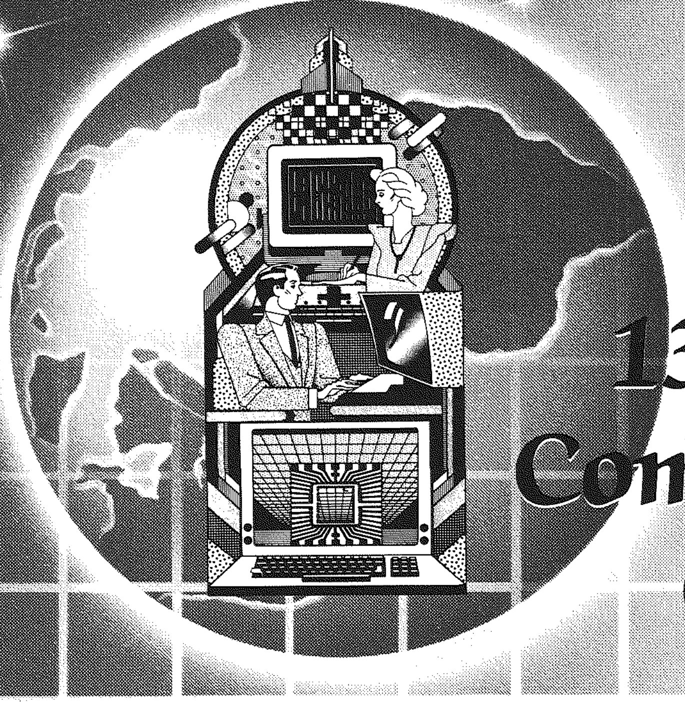
NATIONAL INSTITUTE of STANDARDS and TECHNOLOGY/
NATIONAL COMPUTER SECURITY CENTER



13th National Computer Security Conference

Omni Shoreham Hotel
Washington, D.C.
1 - 4 October, 1990

Proceedings

VOLUME II

Information Systems Security:

Standards - The Key to the Future

Welcome!

The National Computer Security Center (NCSC) and the National Computer Systems Laboratory (NCSL) are pleased to welcome you to the Thirteenth Annual National Computer Security Conference. We believe that the Conference will stimulate a vital and dynamic exchange of information and foster an understanding of emerging technologies.

The theme for this year's conference, "Information Systems Security: Standards -- The Key to the Future," reflects the continuing importance of the broader information systems security issues facing us. At the heart of these issues are two items which will receive special emphasis this week -- Information Systems Security Criteria (and how it affects us) and Education, Training, and Awareness. We are working together, in the Government, Industry, and Academe, in cooperative efforts to improve and expand the state-of-the-art technology to information systems security. This year we are pleased to present a new track by the information security educators. These presentations will provide you with some cost-effective as well as innovative ideas in developing your own on-site information-systems-security education programs. Additionally, we will be presenting an educational program which addresses the automated information security responsibilities. This educational program will refresh us with the perspectives of the past, and will project directions of the future.

We firmly believe that security awareness and responsibility are the cornerstone of any information security program. For our collective success, we ask that you reflect on the ideas and information presented this week; then share this information with your peers, your management, your administration, and your customers. By sharing this information, we will develop a stronger knowledge base for tomorrow's foundations.

JAMES H. BURROWS
Director
National Computer Systems Laboratory

PATRICK R. GALLAGHER, JR
Director
National Computer Security Center

Conference Referees

Dr. Marshall Abrams	*The MITRE Corporation*
James P. Anderson	*James P. Anderson Company*
Jon Arneson	*National Institute of Standards & Technology*
Devolyn Arnold	*National Computer Security Center*
James Arnold	*National Computer Security Center*
Al Arsenault	*National Computer Security Center*
Victoria Ashby	*The MITRE Corporation*
Elaine Barker	*National Institute of Standards & Technology*
Dr. D. Elliott Bell	*Trusted Information Systems, Inc.*
Greg Bergren	*National Computer Security Center*
James Birch	*Secure Systems, Incorporated*
Earl Boebert	*Secure Computing Technology Corporation*
Dr. Dennis Branstad	*National Institute of Standards & Technology*
Dr. John Campbell	*National Computer Security Center*
Dr. Steve Crocker	*Trusted Information Systems, Inc.*
Dr. Dorothy Denning	*Digital Equipment Corporation*
Donna Dodson	*National Institute of Standards & Technology*
Greg Elkmann	*National Security Agency*
Ellen Flahaven	*National Institute of Standards & Technology*
Dan Gambel	*Grumann Data Systems*
Dain Gary	*Mellon National Bank*
Bill Geer	*National Computer Security Center*
Virgil Gibson	*Grumann Data Systems*
Dennis Gilbert	*National Institute of Standards and Technology*
Irene Gilbert	*National Institute of Standards and Technology*
Dr. Virgil Gligor	*University of Maryland*
Capt James Goldston, USAF	*National Computer Security Center*
Dr. Joshua Guttman	*The MITRE Corporation*
Dr. Grace Hammonds	*AGCS, Inc.*
Douglas Hardie	*Unisys Corporation*
Ronda Henning	*Harris Corporation*
Jack Holleran	*National Computer Security Center*
Jim Houser	*National Computer Security Center*
Russ Housley	*XEROX*
Dr. Dale Johnson	*The MITRE Corporation*
Carole Jordan	*Defense Investigative Service*
Sharon Keller	*National Institute of Standards & Technology*
Leslee LaFountain	*National Computer Security Center*
Steve LaFountain	*National Computer Security Center*
Paul Lambert	*Motorola GEG*
Carl Landwehr	*Naval Research Laboratory*
Robert Lau	*National Computer Security Center*

Dr. Theodore Lee	*Trusted Information Systems, Inc.*
Nina Lewis	*University of California, Santa Barbara*
Steve Lipner	*Digital Equipment Corporation*
Terry Losonsky	*National Security Agency*
Dr. Vic Maconachy	*National Security Agency*
Barbara Mayer	*Trusted Information Systems, Inc.*
Frank Mayer	*Sparta*
Vin McLellan	*Boston University*
Catherine A. Meadows	*Naval Research Laboratory*
Dr. Jonathan Millen	*The MITRE Corporation*
William H Murray	*Independent Consultant*
Eugene Myers	*National Computer Security Center*
Ruth Nelson	*GTE*
Dr. Peter Neumann	*SRI International*
Steven Padilla	*Trusted Information Systems, Inc.*
Nick Pantiuk	*Grumann Data Systems*
Donn Parker	*SRI International*
Rich Petthia	*Software Engineering Institute*
Dr. Charles Pfleeger	*Trusted Information Systems, Inc.*
Jerrold Powell	*Department of the Treasury*
Maria Pozzo	*University of California, Los Angeles*
Michael Rinick	*Central Intelligence Agency*
Ken Rowe	*National Computer Security Center*
Prof Ravi Sandhu	*George Mason University*
Marv Schaefer	*Trusted Information Systems, Inc.*
Dr. Roger Schell	*GEMINI*
Dan Schnackenberg	*Boeing Aerospace*
Miles Smid	*National Institute of Standards & Technology*
Suzanne Smith	*Los Alamos National Laboratory*
Brian Snow	*National Security Agency*
Prof. Gene Spafford	*Purdue University*
Dr. Dennis Steinauer	*National Institute of Standards & Technology*
Freddie Stewart	*ANSER*
Dr. Cliff Stoll	*Harvard - Smithsonian Center for Astrophysics*
Marianne Swanson	*National Institute of Standards & Technology*
Mario Tinto	*National Computer Security Center*
Ann Todd	*National Institute of Standards & Technology*
Eugene Troy	*National Institute of Standards & Technology*
LTC Ray Vaughn, USA	*National Computer Security Center*
Grant Wagner	*National Computer Security Center*
Jill Walsh	*INCO, Inc.*
Wayne Weingaertner	*National Computer Security Center*
Roger Westman	*INCO, Inc.*
Roy Wood	*National Computer Security Center*

Thirteenth National Computer Security Conference
October 1-4, 1990
Washington, DC

Table of Contents

VOLUME I

TRACK A - Research & Development

TRACK B - Systems

VOLUME 2

TRACK C - I - Management & Administration

TRACK C-II - Management & Administration

Educator Sessions

Alternate Papers

Student Papers

Special Reprint *12th National Computer Security Conference*

DISASTER RECOVERY / CONTINGENCY PLANNING

Eileen S. Wesselingh

Manager, Consulting Services
National Computer Systems Contingency Services
1250 Northmeadow Parkway
Roswell GA 30076

Focus on Disaster Recovery and Contingency Planning for business survival and continuity has dramatically increased since 1989 - THE YEAR OF DISASTERS. ***EXXON VALDEZ, HURRICANE HUGO, HURRICANE JERRY, SAN FRANCISCO EARTHQUAKE, TORNADOES, FLOODS, EXTREME COLD WEATHER*** - all of these events captured and dominated our attention and interest during the year. We saw and read about untold numbers of heroic people and heroic deeds. We learned that billions of dollars in damages and losses occurred; that lives were lost, that many businesses were destroyed or so severely impacted that they would suffer long term, perhaps even irrevocable, impairment. We were told about the future environmental consequences and problems that would result. But, to too many of us, these tragic events seemed distant and somehow removed from our reality, our worlds. A false sense of security that was shattered on January 15th of this year, when the AT&T long distance lines went down for most of the day. AT&T originally stated that a software bug caused the problem. Latest information now states that it was a worm (a type of computer virus) attack caused by a group of hackers known as The Legion of DOOM.[1] No estimates were ever given regarding overall dollar loss either to AT&T or to the untold number of telecommunication dependent businesses that were affected.

Disasters occur at the local, in your own backyard, level as well. On the weekend of June 15-17, 1990 more than 100,000 homes and businesses, including hospitals and other sensitive services, in Atlanta, GA were affected by a multiple water main break. The breaks were repaired and service was restored by midday Saturday. However, another serious and more far reaching problem resulted that affected the entire metropolitan area until the following Monday morning. Due to lack of water (needed for computer room chillers) the ATM network had to be shut down. People could only access their accounts and/or funds at their own bank. Not only were these people severely inconvenienced, but merchants throughout the area suffered revenue losses. And while this was occurring locally, national attention was focused on the MEGABORG tanker fire off the coast of Texas and severe flooding in the Mid West.

Additional national statistics further give evidence to the increasing fragility and interdependency that exists as we continue to evolve into an automated information dependent economy:

- Every five minutes, a business catches fire in the U.S.; of these 90% suffer catastrophic losses; 40% never reopen.

- 50% of all computer dependent businesses that experience a disaster and do not reestablish processing and operations within 10 days NEVER recover or file for Chapter 11.

- 93% of all businesses which experienced a major Data Processing disaster were out of business in 5 years. Survival odds of 7 in 100.
- An estimated 75% of all data centers in the U.S. store backup tapes **on-site**.
- A business can expect to lose approximately 10% of total gross sales within a week, if data processing is non-operational.

Nine inches of rain in an eight hour period in Chicago on Aug. 13, 1987 seriously damaged dozens of data processing departments. Many issued alerts to backup and hot-site service companies. Five actually declared disasters and went to hot-sites. Initial damage estimates to area businesses were put at over $77 million.

When the Southern Illinois Bell switching station burned on Mother's Day, May 8, 1988, communications over hundreds of thousands of local lines, local fiber circuits, and leased computer lines were brought to a total standstill. Air traffic control was affected, not only at O'Hare, but around the country. The Federal Reserve had to set up a "satellite" office in a car, equipped with a cellular phone, in various shopping center parking lots in order to conduct business with area banks. One company, in the area, activated its contingency plan and hot site. The cost to operate in contingency mode for over two weeks was over $500,000. However, the company estimates that it would have lost $30 million in sales if it had not committed time and resources to develop and implement contingency planning and disaster recovery for the organization.

Natural disasters are not the only risk to be considered. Human error and computer crime are the silent, hidden factors that can cripple or corrupt business. Computer crime losses total over $3 million per year, in the U.S., and are rarely detected. Viruses used to be something that sent us to the doctor's office or home to bed for a week. Today, the word has become part of computer terminology. Computer viruses exist as an ever present threat to any automated business, from mainframe to PC; locally, nationally, and globally. In 1989 and 1990, federal prosecutors obtained convictions in four nationally recognized cases involving illegal access of systems, software theft, destruction of information, and viral attacks.

All these statistics and events clearly give credibility to one very important reality in an increasingly data processing driven business world today: Disasters do happen, and contrary to popular belief or expression, they don't just happen to someone else. Disasters don't discriminate. Disasters aren't self-limiting. Disasters don't call ahead. Accepting and understanding the overwhelming and widespread consequences of having to deal not only with business recovery, but more importantly with BUSINESS SURVIVAL, after a natural or man-made disaster occurs, is becoming one of the most relevant and prominent issues in corporate boardrooms, government and regulatory agencies, management meetings. Concern about this issue even plays an ever increasing role in the normal sales cycle in any automation driven business today. The critical need for a plan, a contingency plan, to ensure recovery and survival, to reduce vulnerability, to protect against liability has become an integral and essential cost of doing business, and stated quite simply, makes good business sense.

In today's information intensive economy, survival of a company may depend on management's ability to use it's information resources to effectively compete, to strategize, to expand and hold market share, in essence to survive.

As information has become more mission critical for normal business operations; as automation has moved from being an after the fact record keeping system to a dynamic, on-line, real time strategic operation, so has the scope of disaster recovery and contingency planning changed from the computer room to the entire business.

Effective contingency planning MUST address the total, integrated business environment, not just the data processing department. Any contingency plan that is not based on this fact will not be workable, will not succeed. Being prepared to deal with a disaster situation, natural or manmade, regardless of the degree, is no longer limited to having a computer to process on and a place to put it. With hot-site and cold-site companies available for contractually committed customers, with quick-ship equipment replacement agreements, with data protection and vault storage services, restoring data processing capabilities can be the least complicated recovery task to accomplish. What degree of business survivability will occur if the DP department is up and running at an alternate site in a few days and the rest of the company is in total shambles? A disaster could occur and seriously cripple or destroy everything but the computer department. It has happened. Or visual another disaster that is occurring with more frequency: A hazardous fire or chemical spill that causes a local or regional evacuation. Your computer room is untouched, your offices are undamaged, your information is not destroyed. The only problem you have is that you can't get to any of it. Perhaps for days or longer. It has happened. Professional journals, newspapers, other forms of media are filled with reports on events, disasters, business interruptions - whatever terminology your organization finds acceptable. However, being unprepared to deal with a disaster and the potential economic and legal consequences resulting from one are no longer acceptable. Businesses develop and implement disaster recovery and contingency planning today not because of the risk that something will happen, but because of the severe economic losses and liability issues that will result if they don't.

Due to the dynamic changes in the financial community caused by deregulation, market aggressiveness, creativity in the investment community and increasing customer sophistication, financial institutions, more than any other types of businesses, are relying more and more on automation in all areas of daily operations to help them better serve their customers. These factors have added increased pressure on financial institutions to develop and incorporate contingency planning and disaster recovery into their business and operational procedures. There is no doubt that collapse of a financial institution would result in severe economic consequences for all business in the area. Additionally, stockholders and customers who demand continuing service will be quick to take their business elsewhere and quick to take legal action.

A recent study done by the University of Minnesota[2] indicates that financial institutions are most seriously impacted by a business interruption with a 1 1/2 day maximum downtime before serious repercussions occur. Federal and state regulators recognized this high risk and have established specific regulations and guidelines in requiring financial institutions to have viable, tested contingency plans in place to avoid the disasters that the loss of information systems and of normal operations, no matter how temporary, can cause. In July of 1989, the FFIEC issued an Interagency Policy On Contingency Planning as a result of a general meeting held in Washington, D.C. in September 1988. This joint policy specifies the scope and level of contingency planning and disaster recovery that it's member institutions must develop and implement. It actually specifies: institution-wide planning, what must be included in the plan,that periodic testing of the plan be performed. Additionally, the FFIEC places responsibility directly on the Board of Directors and senior management for establishing the plan and for annually reviewing, testing, and approving the contingency plan. This annual process must be documented in the board minutes. Each member agency then re-issued the policy under it's own numbering system. The Office of the Comptroller of the Currency reissued it as BC-177, which was first issued in 1983 and applied only to bank's data processing department. Two additional policies, established by the OCC in 1988, extend requirements and responsibilities even further. BC-226 issued on January 30, 1988, sets forth OCC Policy on the end user PC environment. BC-229 issued on May 30, 1988, sets forth OCC Policy in regards to the establishment of Information Security Policies in federal financial institutions.

While financial institutions have been explicitly mandated to develop disaster recovery capability and contingency planning, other industries have varying levels of regulatory guidelines, if any at all. However, there are several laws that do place responsibility for corporate survivability directly on the officers and managers of the corporation. The Foreign Corrupt Practices Act of 1977 states that officers, managers, representative agents of a company can be held criminally liable for failure to safeguard corporate assets in the event of a severe business interruption or failure. (Information is legally defined as a corporate asset.) This liability can result in a personal fine of up to $10,000 and up to 5 years imprisonment. Corporate after-tax penalties start at $1 million. Third party litigation awards could be monstrous. Additionally, directors and officers, when acting for a corporation, are in positions of trust. If they violate that "Duty of Trust", they can be held liable under the "Prudent Man Rule", which requires that they perform their duties with diligence and care. They have a "duty of care" including the obligation to assess the risk of loss and the impact of any loss to the corporation. Directors and officers are also liable under ordinary principles of agency law to investigate and be informed about the condition of the corporation and it's assets.

There are, of course, many other reasons for contingency planning than just the fear of being sued: to protect a business's most valuable assets - it's employees; to protect lives, jobs, assets, market share, to maintain the confidence and trust of employees, stockholders, customers. The reasons for and the necessity of a viable, workable contingency plan are undeniable. But, what exactly is contingency planning? And, what does your organization, business have to do to get one? Before we start discussing the definitions and components involved, three very significant and fundamental caveats MUST be understood and accepted as gospel for your contingency planning efforts to succeed and be workable when needed:

1. It must address the total, integrated business and not only data processing. The plan must belong to entire organization. Everyone must have ownership, but the primary responsibility for creating, testing, maintaining the plan must become part of, or someone's primary job description.

2. Contingency planning is a continuing, on going process. Your business is dynamic not static, and your plan, as an integral part of your overall business, cannot be static either. The contingency process once begun, does not end.

3. There **must** be Corporate Commitment. Without it, you won't succeed. Time, money, personnel, resources will be involved, and not just on a one time basis. If you don't already have it, get it. How? Make upper management aware of the need, and of their responsibilities and their potential liabilities. Use the information in this article for starters. Find out what other companies in your area, type of business are doing. Contact professional organizations for more information. Become the inhouse contingency planning expert.

WHAT IS CONTINGENCY PLANNING?
Simply, it is the identification prior to a business interruption or disaster, of the critical procedures, functions, and resources (personnel, vendors, equipment, supplies, alter-

nate sites, communication needs, etc.) necessary for business survival, not for "business as usual", for SURVIVAL. From this assessment and evaluation process, an action plan is developed to support these procedures. This action plan is a combination of written instructions for taking specific actions when a disaster occurs and skilled, trained personnel functioning as the disaster recovery teams performing these actions. Without a plan, the guaranteed response will be panic and confusion.

WHAT ARE THE STEPS INVOLVED?

1. Upper management must first establish the corporate policies and objectives of the plan: for example: To protect personnel and assets, to continue business operations in a cost-effective manner, to maintain confidence. Identifying these will be critical in deciding on what type of alternate backup site will be needed, i.e., hot-site, cold-site, branch office, new building. If upper management specifies that operations and processing must resume within 24 hours, a hot site must be incorporated into the plan. If operations and processing can be handled manually for one month, other solutions are viable.

2. Personnel must be identified. A Disaster Recovery Coordinator must be named to be responsible for the contingency planning process to which your organization has committed. This person should have certain qualifications: be familiar with the organization, be open-minded and flexible, be persistent and analytical, be a strong communicator and a strong leader. This person will be the driving force for entire process from initiation to ongoing testing, training, maintenance. Disaster recovery teams and team leaders must be identified based on the size and structure of your organization. These will be personnel who are knowledgeable, trained, proficient at what they do. There are three basic types of teams to establish: management - to plan, direct and control, i.e., the Disaster Recovery Team; operational --- the ones who actually implement recovery , for example: Hot-site backup team, user input team, primary site salvage and restoration team; support - to provide all the resources and assistance needed, for example: transportation team, insurance and personnel teams. The number of teams and team members will depend on the size of your organization and the number of sites included. The teams will meet and identify the steps they will perform during a disaster. External organizations/resources must be identified, for example: critical vendors, clients, auditors, consultants, agents. Consultants are excellent, experienced, non-political resources to invest in and utilize. They've been there before. If your organization decides to use one, you will still have to work with the consultant to gather the information, identify the teams, etc. You just won't have to re-invent the wheel.

3. A realistic schedule must be established. Expect nine months to one man-year of personnel time and possibly more to the first draft stage, depending on the size of scope of your plan and the resources available. Several excellent PC based planning software packages are now available on the market. Most of them use relational databases, allow autoload and download of information with other computers, are network compatible, and have other features which should be investigated. These PC based tools allow ease of input, flexibility in planning, and make maintenance doable. It's important to remember that they are a tool to be used in conjunction with, not in place of the organizations efforts to develop and implement survivability.

4. Data gathering. This is the fun part. Sometimes the battles that are waged in identifying and defining the critical procedures and resources are reason enough to declare a disaster. Procedures and functions can be classified into three categories:

   Priority 1 - Those that run the business.
   Priority 2 - Those that control the business.
   Priority 3 - Those that supplement, are informational.

It is critical to realize that depending on the time of the year the priority levels can and will change. Once this prioritization is completed, you can then identify the required resources to support your recovery operations - supplies, forms, equipment, office space, off-site storage requirements (REMEMBER TO STORE A COPY OF YOUR CONTINGENCY PLAN OFF-SITE), personnel requirements, notification procedures and lists, control centers, meeting and alternate work sites, alternate processing sites (redundant, hot site, relocatable shell), telecommunication recovery needs (dial up lines, leased lines, T1's, satellite links, electronic vaulting).

Yes, it is a lot of work, but, one of the benefits will be a tremendous reduction of conflict and disorganization if an actual disaster occurs.

5. Writing the plan. Having approached this process from a "worst possible case" what you will actually have is a series of subsets or "mini-plans" that can be individually or collectively put into action depending on the level and severity of the disaster. Some of the PC based planning software packages I mentioned earlier allow you to do this. Keep it simple. Keep it well organized.

No, you're not finished. You now have this wonderful document, this committed and enthusiastic organization. What's next?

TESTING

How else will you know if your plan will work? If your team members are correctly identified and trained? If the information in your plan is accurate, sufficient? How else will the teams gain experience in emergency actions and recovery procedures? How else will you identify needed changes or additions, problems, weaknesses? How else will you show your organization's commitment to the contingency planning process and keep upper managements support?

Don't be afraid to test. In my book, NO TEST IS A FAILURE. Every one is a learning experience, a chance to improve, and if there are problems, an opportunity to work them out, correct them before a real disaster occurs. Plan your testing. Start small. Don't pull a full scale "mock disaster" test the first time you test. Test a single team or section of the plan. Identify the goals and parameters of the test beforehand. Monitor and audit the actual test. After it's over have a post-test review to determine if you met the objectives and if changes have to be made. Next time you test integrate two or more teams or sections. Do the same pre and post test planning and review. Then when your contingency organization has familiarity and experience with the plan, do the mock disaster. Just be careful not to cause a real one in the process. Think of a test as a dress rehearsal, an opportunity to see how and where the script has to be modified. Testing will lead to maintenance, revisions, changes to your plan. Testing and maintenance will be the forces that keep the commitment to contingency planning and disaster recovery alive and well in your organization.

IMPLEMENTING YOUR PLAN

The infamous middle of the night phone call occurs. There is a problem which may or may not require contingency plan activation. Briefly, let's review the steps major steps or segments of a recovery operation that would take place:

1. Notification. Some type of interruption has occurred. Because your infrastructure is in place, the notifying party contacts predetermined Disaster Recovery Coordinators or Team Leaders.

2. An initial situation assessment is made to determine current status, further steps, back to normal times.

3. Based on their assessment a strategy is selected (Exs: Scale down operations & wait for repair; Declare a disaster and activate plan.)

4. Recommend course of action to Senior management, who has responsibility & authority to activate plan & declare.

5. If a disaster is declared, teams notified, alternate sites notified, other notifications occur.

6. Teams meet at predetermined sites and begin assigned recovery tasks.

7. Contingency working environments (data processing and nondp) are established.

8. Evaluation and restoration occurs at primary site.

9. Planned switchback occurs after primary site problem has been problem.

10. Recovery efforts are reviewed. Contingency Plan and organization are changed and updated to reflect needed changes, if any.

This is a very simplified overview of how the recovery would flow. In actuality, there would be many, many subsets, tasks, and other identified steps and parameters. But they would all generally fit into the broad overview steps above.

Contingency planning and disaster recovery are essential costs of doing business today, of staying in business today. They're here to stay. Not only your business, but the lives of many people --- employees, stockholders, customers, other businesses --- will be affected and impacted by how you address contingency needs and by how you decide on solutions in a viable, tested contingency plan and organization.

References:

1. R. Cringely **"Notes from the Field"**, INFOWORLD, June 18, 1990.

2. University of Minnesota, **"Data Processing Downtime"**, 1982.

Executive Summary

Disaster Recovery of $138 Million Fire

Lloyd R. Smith, Jr.
Colonel, USAF *(retired)*

Information Systems Integrity
Box 95773
Oklahoma City, OK 73143
(405) 737-8348

The Oklahoma Air Logistics Center (ALC), the states largest industrial complex and largest employer, experienced a 40 hour fire that destroyed 17 acres of roof over the maintenance facility before being extinguished through the efforts of 22 fire departments. A disaster recovery operation resulted in 265 computer and peripheral cabinets, as well as 10,000 data tapes and 50 disk packs, being evacuated from three threatened data centers, adjacent to the burning facility. The information systems components were then returned and operations recovered in minimum time.

Lloyd Smith, Director of Information Systems during disaster recovery operation, will discuss the organization, its mission, operational requirements, and contingency philosophy. A video tape of the $138 million fire will be shown. The speaker will then describe the recovery of the data centers, and discuss the lessons learned. Included in this presentation will be data on how to make information systems more secure and sustainable from disasters. This insight was gained through years of operational experience as well as visits to numerous other data centers.

Executive Summary

Plans and Assistance

Jon H. Arneson

National Institute of Standards and Technology

The panel will discuss the computer security plans required of each agency. In addition, it will address the development of a comprehensive computer security program within Federal agencies consistent with the Computer Security Act of 1987, OMB Circular A-130, "Management of Federal Information Resources," and OMB Bulletin 90-08, "Guidance for Preparation of Security Plans for Federal Computer Systems that Contain Sensitive Information." Panel members will include representatives from the National Security Agency, the Office of Management and Budget, the President's Council on Integrity and Efficiency, the National Institute of Standards and Technology, and two Federal agencies.

HARMONISED CRITERIA FOR THE SECURITY EVALUATION OF IT SYSTEMS AND PRODUCTS*

A. Brouwer[1], P. Casey[2], D. Herson[3], J. Pacault[4], F. Taal[5], U. Van Essen[6]

ABSTRACT

This paper sets out the technical approach adopted for the Information Technology Security Evaluation Criteria (ITSEC), the harmonised security evaluation criteria produced by France, Germany, the Netherlands and the United Kingdom. It discusses their structure and nature, and the rationale underlying their development.

INTRODUCTION

Within Europe there have been a number of national initiatives in the development of Information Technology (IT) security evaluation criteria, some of which have resulted in publicly available documents, such as the IT-Security Criteria produced by ZSI, the German Information Security Agency [1], the UK Department of Trade and Industry (DTI) Commercial Computer Security Centre's "*Green Books*" [2] and the UK Communications-Electronics Security Group's Memorandum Number 3 on UK System Confidence Levels [3]. There have been other European national initiatives, but their results have not been published or are not widely available.

Recognising common interests, and in order to assist in the development of the market for IT security products, a number of European countries (France, Germany, the Netherlands and the United Kingdom) have been co-operating in the development of a single set of harmonised criteria for the security evaluation of IT products and systems, for possible adoption within the practical evaluation and certification schemes planned or operating in those countries and, ideally, on a wider international basis. These criteria - the Information Technology Security Evaluation Criteria (ITSEC) - were published in draft form in May 1990 [4], and have been widely distributed for comment and consultation, both within the four countries that contributed to their development, and beyond. The Commission of the European Community sponsored a two day consultative conference on the criteria in Brussels during September.

* This paper may be copied freely provided that the source is acknowledged. Please address correspondence to the Computer Security Branch, Information Technology Division, Department of Trade and Industry, Room 847, Kingsgate House, 66-74 Victoria Street, London SW1E 6SW, United Kingdom.

[1] Ministry of the Interior, The Hague, The Netherlands

[2] Department of Trade and Industry, London, United Kingdom

[3] Communications-Electronics Security Group, Cheltenham, United Kingdom

[4] Service Central de la Sécurité des Systèmes d'Information, Paris, France

[5] Netherlands National Communications Security Agency, The Hague, The Netherlands

[6] German Information Security Agency, Bonn, Germany

The harmonised criteria are intended to address the requirements of both the commercial and government security market places, and to be applicable both to individual systems and to IT products designed for sale as components of secure systems. In order to encompass this wide field of applicability, the criteria separate the specification of the security functionality of an IT system or product from the assessment of the assurance that can be held in that functionality. Similarly, the assurance criteria distinguish between the confidence that can be held in the correctness of the implementation of security functions and confidence in their effectiveness in operational use as countermeasures to the actual threats to security that may exist in the context of a particular IT system and its environment.

Wherever possible the ITSEC has drawn on the results of the existing national European initiatives, selecting the most suitable existing approach or method where divergent alternatives existed. However, given the wide intended scope of the ITSEC, it has inevitably been necessary to develop some new ideas to cover areas where no existing approach was satisfactory.

SPECIFICATION OF SECURITY FEATURES

Within the ITSEC, the term **Target of Evaluation** (TOE) is used to refer to a particular IT system or product which is the subject of a security evaluation. In National Computer Security Center terms [5], this could be either a product evaluation or a certification evaluation. In either case, before evaluation can take place, it is necessary to have a precise specification of the features of the TOE which contribute to security. This is done through the definition of a **security target**.

The document or documents which specify the security features define the security target for the TOE and are used as a baseline for evaluation. The security target must always specify the security functions of the TOE. It may specify its security objectives (including information on known threats against those objectives) and it may also specify particular security mechanisms which are to be employed.

In the case of a product, the intended environment and security objectives of a purchaser are not necessarily known: ultimately each purchaser must judge whether a particular product will fit his system security objectives. However, the ITSEC requires that some rationale for the provision of the security functions within a product is given, to provide a context for evaluation and to guide prospective purchasers. Similarly, the developer must also document all assumptions made about the intended environment and the way that the product will be used.

As well as the definition of functionality, the security target must also claim a targeted evaluation level, and a minimum strength rating for the ability of the security mechanisms of the TOE to withstand direct attack.

SPECIFICATION OF SECURITY FUNCTIONALITY

The most important part of a security target is, of course, the definition of the security functions to be implemented by that TOE. The ITSEC recommends eight generic headings under which functions should be grouped:

Identification and Authentication - functions to establish and verify the claimed identity of a user;

Access Control - functions to control the flow of information between, and the use of resources by, users, processes and objects, including the administration and verification of access rights;

Accountability - functions to record the exercising of rights to perform security-relevant actions;

Audit - functions to detect and investigate events that might represent a threat to security;

Object Reuse - functions to control the reuse of data objects;

Accuracy - functions to ensure the correctness and consistency of security-relevant information;

Reliability of Service - functions to ensure consistency and availability of service;

Data Exchange - functions to ensure the security of data during transmission over communications channels.

This approach to grouping functionality is similar to that adopted by the TCSEC Subsystem Interpretation [5].

There are no restrictions on the functionality that can be specified in a security target. The ITSEC predefines some standard functionality classes, but functionality can also be stated explicitly, or as a combination of functionality classes and additional explicit functions.

The ten predefined functionality classes are identical in content to those defined within the German National Criteria [1], but have been restated using the generic headings listed above. The first five correspond to the functionality contained within the assurance classes defined by the US Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) [6]. F1 corresponds to the functionality of US Class C1, F2 to C2, and so on up to F5, which corresponds to US Class B3. (US Classes B3 and A1 possess the same functionality, thus F5 also corresponds to US Class A1.)

The remaining five classes are intended to match common requirements for particular types of system. F6 is intended for use where high data or program integrity is required. F7 is intended for use where high availability is necessary. F8 specifies high requirements with regard to the maintenance of integrity during data exchange. F9 provides high confidentiality during data exchange. F10 is intended for networks with high requirements for both confidentiality and integrity of the information transmitted over the network. It is expected that other predefined classes will be added with time to meet market needs as requirements evolve.

The ITSEC identifies three types of specification style that may be used to express security functions within a security target: informal, semi-formal and formal. Acceptable styles vary, depending on the targeted evaluation level.

An example of an informal style is natural language (ie. the unrestricted normal written or spoken use of a language such as English). Semi-formal styles use some form of restricted notation, in accordance with a set of conventions that reduce potential ambiguity, but which are not formally or mathematically defined. An example is the Claims Language defined within the ITSEC, which has been developed from the Claims Language proposed within the DTI draft criteria [2].

A formal style requires a mathematically based notation. Such specifications can be tested for ambiguity, and can also be shown to be consistent, and correct with respect to a set of axiomatic properties. Special training and aptitude is normally required to produce and use formal specifications, and their correct meaning is often not intuitively obvious. Examples of formal specification styles are formal specification languages such as GYPSY [7] or Z [8] (for readers unfamiliar with Z, [9] provides a example of its use from the standard American literature), or the application of established formal security policy models which reflect the desired security properties of the TOE.

ASSURANCE

As has already been said, the assurance criteria of the ITSEC distinguish between the confidence that can be held in the correctness of the implementation of the security functionality of a TOE and the confidence that can be held in its effectiveness in operational use. There is no required link between the functionality specified and the level of assurance that is claimed.

During the course of evaluation the gathering and examination of evidence for the assessment of correctness and effectiveness are unavoidably intertwined. However, in order to determine whether the required evaluation rating has been achieved, an assessment of correctness is made first, by confirming that all the correctness criteria of the ITSEC defined for the targeted evaluation level have been satisfied. Only once correctness has been established is assessment made of the effectiveness of the TOE, using criteria which apply to a TOE viewed as a coherent whole.

A successful evaluation will confirm that the targeted evaluation level has been achieved and that the claimed minimum strength rating for the security mechanisms is justified.

CORRECTNESS CRITERIA

The ITSEC defines seven evaluation levels in respect of the correctness of a TOE. E0 designates the lowest level and E6 the highest.

Level E0 represents inadequate assurance, the equivalent of TCSEC Division D. A TOE assigned a confidence level of E0 for correctness need not be considered from the point of view of effectiveness, since it has already been shown to be inadequate.

Remaining levels require satisfaction of progressively more stringent criteria in respect of phases or aspects of the development process, development environment, operational documentation and operational procedures for the TOE.

The development process is considered to be made up of a number of phases which take place in turn. Factors contributing to the development of confidence are identified in the criteria for each phase of the process. The four identified phases are:

> **Requirements** - the identification and description of the security target for the TOE;
>
> **Architectural Design** - the overall top level definition and design of the TOE, identifying its basic structure, its external interfaces and its separation into major software and hardware components;
>
> **Detailed Design** - the refinement of the architectural design to a level of detail that can be used as a basis for programming and hardware construction;
>
> **Implementation** - the translation of the detailed design into actual hardware and software, and the testing of this implementation of the TOE against its specifications.

This explicit model of the development process is intended to simplify identifying the necessary relationship between the security criteria and the normal documentation and procedures used in any development performed in accordance with good quality and software engineering practices. Indeed, the ITSEC model is close to the spirit of US DoD software development standard DOD-STD-2167A [10].

The criteria for the development environment address a number of issues relating to the working practices of the developer of the TOE. Particular emphasis is placed on configuration control, the choice of programming languages and tools, and the developer's own internal security measures.

The criteria for operation distinguish between operational documentation (used by the developer to communicate information about the TOE to his customers) and information concerning the operational environment of the TOE. In the case of a system which is already in use, it is possible to assess actual operational procedures. In other cases, it is only possible to evaluate proposed or suggested procedures, set down either by the developer or by the user.

Each of the six levels E1 to E6 specifies evaluation criteria in respect of the defined phases or aspects of development and operation. For each level, the documentation that must be provided for examination is identified, followed by requirements for its content and presentation or for the procedures and standards it must define. These requirements are followed by a definition of the evidence necessary to show that the criteria in question have been met. Finally the actions to be performed by the evaluator are stated.

For clarity, since there are significantly different requirements for each evaluation level, the criteria for each level are set out separately within the ITSEC. There is a need for greater rigour and depth in the evidence required at higher evaluation levels. However, in general each level retains the criteria of the previous lower level, but adds greater detail to the evidence required and also adds new criteria to be considered.

Except at E1, the burden for the provision of evidence is placed on the sponsor (the person or organisation requesting evaluation), who must in turn obtain it from the developer. The evidence provided is then checked or audited by the evaluator. The evaluator is only required to generate evidence where independent production is considered necessary to provide the necessary level of confidence in its conclusions.

For example, there are requirements to provide evidence of functional testing placed on both sponsor and evaluator. The major requirement is for the sponsor to show evidence of comprehensive testing, in particular test plans and test results, produced as part of his normal development practices. The requirement placed on the evaluator is to show that he has examined the results provided by the sponsor, checked their comprehensiveness and accuracy by performing limited testing of his own, and repeated any tests where he found points of apparent inconsistency or error in the results provided.

At Level E1, and only that level, it is acceptable that where the sponsor of evaluation cannot obtain adequate evidence from the developer, it may be generated by the evaluator. At higher levels this is considered to compromise the independence of the evaluator to an extent where adequate assurance could not then be held in the evaluator's conclusions.

This approach gives greater detail of the roles of developer and evaluator than has been found in previous evaluation criteria. This helps ensure uniformity of evaluations, assists developers to identify before development starts the information that must be produced and retained for use by the evaluators and, last but not least, protects developers from over-zealous evaluators.

EFFECTIVENESS CRITERIA

Evaluation of effectiveness takes account of the proposed use of the TOE in the context of its intended environment. It is only performed after confidence in correctness has been established. Because these criteria are applied to the TOE as a whole, they are not broken down by evaluation level, or by phases and aspects of construction and operation.

Six distinct aspects of the TOE are considered:

Suitability of Functionality - whether the TOE's security functions will counter the identified threats against the TOE as specified in its security target;

Binding of Functionality - whether the individual security functions and mechanisms of the TOE work together in a way which is mutually supportive and coherent;

Assessment of Construction Vulnerabilities - whether vulnerabilities within the construction of the TOE identified during the course of evaluation will be exploitable in practice;

Strength of Security Mechanisms - an assessment of the ability of the TOE to withstand direct attack on its security mechanisms;

Ease of Use - the practicality of the security functions and mechanisms of the TOE for actual live operation;

Assessment of Operational Vulnerabilities - whether vulnerabilities within the operation of the TOE identified during the course of evaluation will be exploitable in practice.

The criteria for each of these aspects have the same format as for an aspect of correctness. Requirements for content and presentation are followed by requirements for evidence, and then evaluator actions are defined. Unlike assessment of correctness, most of the work in assessment of effectiveness is performed directly by the evaluators. Most of the necessary evidence will have been already been obtained during the evaluation of correctness.

As part of the documentation required for evaluation of correctness, the sponsor of evaluation will have supplied a security target for the TOE. During the evaluation of correctness, this target will have been examined for coverage and consistency. When assessing **Suitability of Functionality** the target is used to determine whether the security functions and mechanisms of the TOE will in fact counter the identified threats to the security of the TOE. If the security target for a product is defined to be one or more predefined functionality classes, without extension or omission, then the TOE is assumed to be suitable for its intended purposes.

Even when a TOE's security functions have been determined to satisfy the suitability criteria, it is possible that the mechanisms employed may interfere or conflict with each other. Assessment of **Binding of Functionality** ensures that the security functions and mechanisms work together in a way that is mutually supportive and provides an integrated and effective whole. It requires analysis of the interrelationships between security functions and mechanisms to show that it is not possible to cause any security function or mechanism to conflict or contradict the intent of another in such a way that an exploitable weakness results. Clearly, effective binding of functionality is a essential property of a Trusted Computer Base (TCB) as envisaged by the TCSEC.

During the assessment of the correctness of a TOE, various vulnerabilities (security weaknesses) will have been identified. During the **Assessment of Construction Vulnerabilities** impact analyses are performed to assess whether the vulnerabilities that have been found would, in practice, actually compromise the security of the TOE as specified in its security target. It is possible that threats that could exploit the identified weaknesses can be shown not to exist in practice, or to be countered by other security mechanisms within the TOE, or to be countered effectively by other means external to the TOE.

Even if a security mechanism cannot be bypassed, deactivated or corrupted, it may still be possible to defeat it by direct attack based on deficiencies in its underlying algorithms, principles or properties. Assessment of the **Strength of Mechanisms** is distinguished from other aspects of evaluation in that it requires consideration of the level of resources that would be needed for an attacker to execute a successful direct attack. The objective of this assessment is to confirm the claimed minimum strength rating for the critical mechanisms of the TOE. A critical mechanism is one that is not protected from attack by other, stronger, mechanisms and whose failure would create a vulnerability.

The minimum strength of a TOE's mechanisms is rated as either basic, medium or high: **basic** provides protection against random accidental subversion, but not against knowledgeable attackers; **medium** provides protection against attackers with limited opportunities or resources; **high** could only be defeated by attackers possessing a high level of expertise, opportunity and resources, with successful attack being judged to be beyond normal practicality.

In certain evaluations it may not be possible or sensible for the evaluators to perform an assessment of the strength of all mechanisms; for example, in the case of cryptographic mechanisms the rating would be supplied by an appropriate national body with the necessary specialist skills and information.

Assessment of **Ease of Use** requires review and analysis of external security measures, such as procedural, physical and personnel controls, that are required to support the security functions and mechanisms of the TOE. The methods of configuration and operation of the security functions and mechanisms of the TOE are also assessed to ensure that they are practicable for operational use.

Assessment of Operational Vulnerabilities is analogous to the Assessment of Construction Vulnerabilities, but investigates weaknesses found in the documentation and procedures for configuration and operational use of the TOE.

A failure of the TOE to satisfy any of the identified aspects of effectiveness will result in the TOE being considered to provide inadequate assurance.

RELATIONSHIP TO THE US TCSEC

An important aspect of the ITSEC must be to enable the results of an evaluation to be compared against the TCSEC. As has already been stated, the predefined functionality classes F1 to F5 are designed to correspond to the functionality of the TCSEC classes. The evaluation levels, however, have been derived by harmonisation of various European IT security criteria schemes and contain a number of requirements which do not appear in the TCSEC explicitly, and make direct equivalence of evaluation levels significantly more difficult.

However, it is possible to produce a crude table showing the intended correspondence between the ITSEC criteria and the TCSEC classes:

ITSEC		TCSEC
E0	--->	D
F1, E2	--->	C1
F2, E2	--->	C2
F3, E3	--->	B1
F4, E4	--->	B2
F5, E5	--->	B3
F5, E6	--->	A1

A product successfully evaluated against the ITSEC for a predefined functionality class F1 to F5 at an evaluation level not lower than that given in the table above should fulfil the requirements of the TCSEC class shown in the table. The converse relationship, however, cannot be directly assumed, due to the wider confidence requirements found in the ITSEC criteria: although any product of sufficient quality to pass a TCSEC evaluation is likely to have been built in a manner that would meet the additional software engineering requirements of the ITSEC, this would have to be proven by evaluation.

In addition, there are other considerations that must be taken into account. Within the TCSEC great importance is placed upon the concept of a reference monitor contained within an identifiable TCB to enforce access relationships. The ITSEC does not mandate particular types of mechanism. It would therefore be necessary for a reference validation mechanism meeting the identified design requirements expressed in the TCSEC to be specified as part of the security target in order for a rating of true equivalence to be possible.

CONCLUSIONS

The ITSEC represents a new development in security evaluation criteria. It has consciously attempted to address a wider field of applicability than previously developed criteria, by addressing both systems and products, and by addressing the needs of both commercial and government market sectors. Equally, it has attempted to draw from the best features of existing criteria, to harmonise where possible and to innovate only where necessary.

The ITSEC has currently been issued in a draft form, and its authors accept that it will inevitably contain errors, inconsistencies and omissions. It is being subjected to wide review and comment on an international basis, in order that after any necessary revision it may receive broad acceptance and adoption by a wide range of users and market sectors.

Comments and suggestions are earnestly invited; contact addresses are given in the ITSEC for each of the four contributing nations. Further copies of the ITSEC can be obtained from the same addresses or via the contact address referenced at the head of this paper.

REFERENCES

1 Criteria for the Evaluation of Trustworthiness of Information Technology (IT) Systems
ISBN 3-88784-200-6, German Information Security Agency (ZSI), Bonn, Germany, January 1989.

2 DTI Commercial Computer Security Centre Evaluation Criteria
Department of Trade and Industry, London, UK, draft February 1989.

3 UK Systems Security Confidence Levels, CESG Memorandum No. 3
Communications-Electronics Security Group, Cheltenham, UK, January 1989.

4 Information Technology Security Evaluation Criteria (ITSEC)
German Interior Ministry (for the Four Nation Group), Bonn, Germany, draft May 1990.

5 Computer Security Subsystem Interpretation of the Trusted Computer System Evaluation Criteria
NCSC-TG-009, National Computer Security Center, Washington, USA, September 1988.

6 Department of Defense Trusted Computer System Evaluation Criteria
DOD 5200.28-STD, Department of Defense, Washington, USA, December 1985.

7 Report on Gypsy 2.05
D. I. Good et al, Report ICSCA-CMP-48, University of Texas at Austin, USA, February 1986.

8 The Z Notation: A Reference Manual
J. M. Spivey, ISBN 0-13-983768-X, Prentice Hall International, 1988.

9 A "New" Security Policy Model
P. Terry et al, Proceedings of the 1989 IEEE Symposium on Security and Privacy, pp. 215-228, IEEE Computer Society Press, USA, 1989.

10 Military Standard: Defense System Software Development
DOD-STD-2167A, Department of Defense, Washington, USA, February 1988.

THE VME HIGH SECURITY OPTION

Tom Parker
Principal Security Consultant.
ICL Defence Systems, Eskdale Road, Winnersh, Wokingham,
Berkshire, England.

Abstract

Criteria for evaluating the security properties of computer systems are now well established and widely accepted. The security capabilities of ICL's VME Operating System have recently been enhanced in accordance with the requirements they lay down at level B1 on the NCSC scale, by the addition of a High Security Option (the VME HSO). This paper describes the VME HSO, concentrating on the features that have enabled it to achieve the equivalent of level B1 certification by the British Government. The product is also aimed at the commercial market, and the paper describes the integrity, audit and usability features that have been provided to satisfy requirements in this area.

1. INTRODUCTION

ICL is one of the leading European manufacturers of computer systems. Its proprietary mainframe operating system is called the Virtual Machine Environment, or VME. The VME Operating System has established itself over the years as being one of the more secure commercial operating systems available. A previous paper (Ref 1) presented at the DoD Conference described the ways in which the software and hardware architectures of VME and its host systems combine together to provide the fundamental structure upon which secure higher level functions can be built.

That paper was presented nine years ago, and VME has since then moved forward a long way. At the time of the paper, ICL was making initial proposals to the British Government's Department of Industry (as it was then called) to develop with their financial support a purchasable set of security enhancements. The development was to be aimed at both the government and commercial user populations; it was to provide usable security, offering a great deal of flexibility in the choice of security policy; it was to provide strong security conforming to recognised independent security quality standards and it was to be evaluated by a recognised authority, resulting in a written certificate that ICL could use in the marketplace. In December 1987 ICL had the go-ahead to develop the product, and work started in 1984.

The development came to be known as the VME High Security Option, usually called the VME HSO. It is available on ICL's Series 39 mainframes.

2. MEASURING HOW GOOD IT IS

In the past, various techniques have been applied in a somewhat ad-hoc manner to assess the quality of security provided by a system. Some examples are:

- straightforward functional testing of security features;
- examination of source code listings, sometimes aided by automatic tools;
- "tiger team" attacks in which security experts attempt to penetrate the system in a simulated real-life situation;
- consideration of architectural and design quality; a well designed and structured system is easier to assess and more likely to be correct;

- formal verification of the security properties of the system using mathematical techniques; such techniques are at present feasible only on small systems designed and written in special languages.

2.1 Standard Evaluation Criteria

Until the 1980's there was no way of obtaining a reliable standard measure from these techniques; there was no concept of marks out of ten or position on a scale of security. However in August 1983 the first official set of standard criteria for the evaluation of computer security was published by the US Department of Defense (Ref 2). This standard is widely applied in US Government procurements of secure systems and its influence is now pervading European government and commercial requirements. Although the evaluation scale has been subjected to criticism relating to its scope of application and its too close coupling of functionality and correctness requirements, it is a remarkable technical achievement. It represents the culmination of a decade of research, and it is the only universally recognised scale available today.

The scale comprises the following set of levels, given in increasing order of security quality:

Level D:	No security.
Level C1:	Basic security for benign users.
Level C2:	Strong conventional security. Discretionary security policy only.
Level B1:	Support of a Mandatory, centrally imposed security policy. Defence against corrupt application code is possible.
Level B2:	Level B1 strengthened by a more rigorous and comprehensive approach and a better security structure.
Level B3:	Specially designed with security as an overriding priority.
Level A1:	Similar to B3, plus formal methods.

A complex set of criteria apply at each level covering the system's design, functional capabilities, testing, development environment and documentation.

An evaluation scale has also been developed by the British Government (Ref 3). This has built on the experience obtained from the American work and is more flexible in its application. One important feature is its ability to separate out questions about what the system does, from questions about how well it does it.

Other scales are emerging from the German Government (Ref 4) and from the Department of Trade and Industry in the UK (Ref 5). A first draft of a harmonized scale has also been published (Ref 9).

Over a period of two years, starting in early 1987, the VME HSO was subjected to an intensive security evaluation by an independent technical team of security experts funded and controlled by the British Government. ICL's aim was to obtain certification for the system at a UK equivalent of level B1 on the American scale, with an similar but more complex rating on the British Government's own scale. This was successfully achieved in May 1989 for version SV221 of the system. ICL is now discussing ways in which this certification can be carried forward into subsequent versions.

At the time of writing this paper, ICL knows of no other proprietary general purpose operating system which currently offers this level of security functionality and assurance.

3. LEVEL B SYSTEMS - WHY SO MUCH BETTER?

The single most significant advance in the transition from a level C to a level B system is the support of a centrally controlled Mandatory Access Control Policy, which provides for data confidentiality without relying on the good behaviour of either end users or the application software they use..

Such a policy possesses three major features:

- it allows a security manager to determine and mark the levels of confidentiality of data held by the system;
- it allows the manager to determine and mark which users are cleared to access what data according to its confidentiality markings; only if a user is "cleared" to the level of confidentiality of the data is he permitted to read it;
- using a rule known as the "information flow" rule it prevents untrusted application code from circumventing the above checks by maliciously copying data from a highly confidential file to a less confidential file which a user with a low clearance can subsequently access. Such code (sometimes called "Trojan Horse" code) may attempt to do this unknown to the user for whom it is executing.

The first two features are relatively easy to provide. The standard approach is to associate a confidentiality label with each data object (its "Classification") and with each user (his "Clearance"). When a user requests to read an object his Clearance is compared with the object's Classification and access is permitted only if the former is higher than the latter.

The third feature however, is at the same time the most significant and the most difficult. Its significance lies in the fact that without it, it is not possible to evaluate only some of the code in a system to be sure of the system's security; without it, all code needs to be evaluated. The latter is one of the major deficiencies of level C systems; the ability to distinguish between "trusted" and "untrusted" code is the big leap forward that level B systems make, both in terms of assurance and in terms of their basic evaluatability. It is only on level B systems that a user can execute some unknown code on his confidential file and be sure that its contents cannot be leaked by that code to any user not permitted to see it.

The protection offered by information flow control is difficult to achieve because its straightforward imposition can be too restrictive in real application environments. An operating system cannot be expected to understand the internal logic of all of the application code that might ever be executed on it, so it must lay down information flow rules in ways that do not depend in any way on such an understanding. These rules will necessarily be restrictive in nature; they will be a blunt instrument that, unless used with great care, might cause legitimate operations to fail because the system cannot be sure of their legitimacy.

An example may help illustrate this point (see Figure 1). Suppose a particular application is performing two "update by copy" operations in parallel. The first involves reading a confidential file CA and copying it with appropriate changes to a second confidential file CB; the second involves reading a publicly available file PA and selectively copying it to a similar public file PB.

Neither of these operations constitutes an information flow violation, but the Mandatory Policy cannot permit the application simultaneously to open CA for read access and PB for write access in case it maliciously, or accidentally copies data from CA to PB. The operating system does not know that the application will not do this, and it certainly does not trust it not to do it. Indeed if the application had been tampered with and contained maliciously written Trojan Horse code, it would be very likely to do something of this kind!

So the problem is not that information flow security is difficult to achieve (at least at B1 levels of assurance), but rather that it is difficult to be sure that applications will still work when information flow controls are imposed. More significantly, the working of the system itself might depend on the

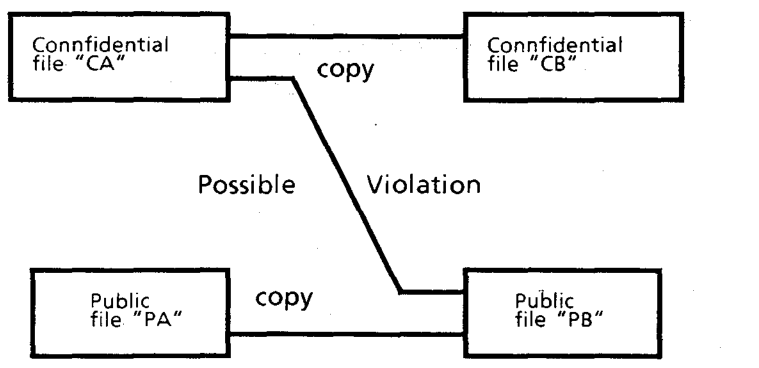


Figure 1: Information Flow Example

continued functioning of standard system "application" processes which might suffer at the hands of the information flow rule.

The next Section describes how ICL solved these problems in VME.

4. USABLE FLOW CONTROLS

Naturally, the VME HSO supports a Mandatory Confidentiality Policy with information flow controls. To do this it uses security labels along the lines already described. The system is designed so that a security manager can choose names for the labels that are appropriate for his needs, and they may be chosen to describe either or both hierarchic confidentiality levels and non-hierarchic confidentiality compartments. The basic flow control rules outlined above are applied, but enhanced in a number of ways which minimise their impact on the system's usability:

- It is possible for a suitably trusted user to mark code so that the Virtual Machine it is executing in is permitted potentially to violate the information flow rule. It is expected that installation management will themselves ensure that no actual flow violations would occur. The legitimate application in the example could be marked in this way to enable it to work, but only after it had been suitably vetted to be trustworthy in this respect . For this reason the marking is said to belong to a group of markings known in VME as "trust" markings. VM's executing in possession of any of these trusts are known as "Trusted Processes". - It is possible for a system administrator to run an application in a way which allows it to continue to work as if no Mandatory Policy controls were being applied, but which audits all cases where violations would have been caused had these controls been in force. In the example application the opening of PB for write access would trigger an audit message because CA had already been opened for read access.

  This information can be used in two ways: either the application can be reorganised to complete and close down its PA to PB copying before opening the confidential files - in which case not even a potential violation is caused and the application can now be run under full controls with no security problems, or the application can be marked as trusted as described above, with the exact reason for the need for trust having been identified and the verification of the code's trustworthiness having consequently been made easier. This "trial" mode method of running an application is intended to be used primarily as a transition aid.

- It is possible for a security manager to configure the Mandatory Policy for the system so that flow control is not policed at all. For many commercial systems who perceive that the threat of information compromise by this means is small this would be an appropriate choice.

- It is possible for a security manager to constrain information flow to be permitted potentially to occur only within a band of confidentiality levels. In the example it could be arranged to permit potential CONFIDENTIAL to PUBLIC information flow, but prevent any flow from, say, SECRET files.

5. INTEGRITY

Although secure commercial organisations have a strong interest in protecting sensitive information from getting into the wrong hands, their major motivation for obtaining and installing a secure system is usually that of prevention of fraud. One of the technical consequences of this is that there is a need to provide data integrity, and it is therefore at least as important for a secure operating system to provide strong data integrity controls as it is to provide strong data sensitivity controls. A particularly significant paper on this topic was recently published by Clark & Wilson (Ref 7).

Because of this importance, despite there being no specific requirement to do so in the American evaluation criteria, ICL has implemented a Mandatory Integrity Policy in the VME HSO, to complement its Mandatory Confidentiality Policy. The integrity features closely parallel the confidentiality features, and are as follows:

- The policy allows a security manager to determine and mark the level of integrity of data held by the system.

- It allows the manager to determine and mark which users are cleared to modify what data according to its integrity level. Only if a user's integrity clearance is higher than the integrity level of the data is he permitted to modify it.

- It prevents high integrity data from being corrupted by low integrity inputs; this rule is an integrity dual of the confidentiality information flow rule. It is Trojan Horse data rather than Trojan Horse code which is defended against here.

There is however an important difference between the two policies. The Mandatory Confidentiality Policy has no interest in distinguishing between different untrusted application code modules that may be used to access highly confidential data; there is no concept of giving code a clearance. The integrity policy does have this concept, and there is a therefore a fourth feature:

- It allows the security manager to determine and mark which code modules are cleared to modify what data according to its integrity level.

By making use of this rule the security manager can be sure that important application data is operated upon only by the proper authorised application code under the control of a properly authorised user. Furthermore, the integrity flow rule prevents such an application being spoofed into running with unauthorised input data.

The VME HSO allows the security manager a high degree of flexibility in the way in which this policy is applied. The enforcement options are similar to but separate from those available for the Mandatory Confidentiality Policy. In particular the integrity information flow rule which prevents the flow of low integrity data into high integrity data can in real situations be relaxed if the code involved has been produced to defend itself against spoof input or other low integrity input, or if other features of the operating environment can be used to guard against supply of the wrong input data.

Finally a word about viruses: high integrity software on a VME HSO system is protected against modification by the same integrity marking as that given to that software when it is executing; in this way the Mandatory Integrity Policy ensures that no software of lower integrity could modify it. This means that a virus, which reproduces itself by copying itself from software module to software module will always be confined only to at most the integrity level of the software module within which it is introduced into the system. The operating system code of the VME HSO is protected by a special integrity label which customers do not use for their own data files. This code is therefore protected against corruption from any viruses that may be unknowingly introduced on customer program files. Similarly, a security conscious customer site, by ensuring that all unknown software is introduced only at a very low integrity level until it has been given a clean bill of health, can guard its own software against viral attack.

6. OTHER MANDATORY POLICY REFINEMENTS

There are a variety of other ways in which it is useful, and sometimes essential to allow the use of a Mandatory Confidentiality or Integrity Policy to be applied to fit a particular installation's needs.

Once the concept of being able to mark code, or nominate users as trusted has been implemented, one can identify a further set of privileged functions that can be controlled using different trust markings. Examples of functions that can be specifically controlled under the trust system of the VME HSO are:

- the ability to change a confidentiality or integrity label,
- the ability to set and change security controls,
- the ability to set and change audit controls,
- the ability to change one's own password,
- the ability to introduce alien magnetic tapes into the system,
- the ability to override the discretionary access control system.

Only users given the appropriate trust, using software marked with the same trust may perform any of the functions listed above. In total there are nearly thirty different categories of trust supported on the VME HSO, allowing a fine degree of tailoring to be applied. Services can also optionally be controlled by trust, and users can be confined to a particular application service when operating with a particular trust.

There are two points of particular interest in these examples.

The first is the distinction that is possible between the audit and security management functions; this distinction allows a clear separation of responsibility to be enforced between these two important roles.

The second is the way in which trusts are used to strengthen the discretionary access controls of the system. There are two operating system commands in VME which have in the past been considered particularly dangerous; these are SWITCH-USER and SET-PERMISSION-OVERRIDES. Either of these permit a user effectively to by-pass discretionary security controls. Naturally the use of these features was carefully controlled under the discretionary access control policy. However it is now much more difficult to penetrate the system by illegally obtaining access to these privileged commands; There are two reasons for this. The first is that their use is now controlled under the Mandatory Policy. Only trusted users may use them, and then of course only subject to the discretionary controls that have always applied. The second is that the power of these privileged commands is significantly reduced in VME HSO systems; even if an untrusted user were to switch to become the Security Manager, he would be unable to exercise the powers of that user since it is only that user's discretionary capabilities that are acquired. The intruder would not have been given either the trusts or the clearances of the username to which he has switched.

The trust system has been extended in other ways. It is possible to nominate particular workstations as possessing or not possessing particular trusts. In this way users can be controlled in their choice of workstation from which they are permitted to perform their trusted actions. This idea is further extended to encompass other communications devices like network gateways and cluster controllers, so it is possible for example to prohibit any trusted activities coming in from a gateway to a public X25 network.

The Mandatory Policy labelling scheme is also extended to cover communication devices and links, so it is possible to label a workstation with a particular confidentiality or integrity clearance which limits the effective clearance of users that use that workstation. For example, an installation could limit work on data with integrity category of PAYROLL to those workstations located in the payroll office.

Similarly, it is possible to mark devices like disc drives and line printers with security labels which constrain them to handling data whose security markings lie between defined boundaries. In this way, the printing of confidential or high integrity data can be confined to one or more nominated printers, and the storage of particularly sensitive files can be confined to nominated disc or tape drives. This latter form of protection can also be applied in terms of disc and tape partitions themselves, permitting fine grain control over file placement.

Finally, it is possible to mark application services with labels so that whatever a user's clearance, for certain types of application service it can be bounded according to the nature of the work being done.

7. AUDITING

The ability to record for posterity what is happening on a system is almost as important as the ability to control it in the first place. Auditors wish to make system users accountable for their actions; they wish to analyse the ways in which a system is being used or abused; they wish to be able to look back to a record of previous events in order to assess damage done when a belated discovery of an attack on the system has been made; they wish to be able to monitor particular kinds of action or action by particular individuals or action using particular system access points in order to forestall attacks on the system; finally they wish to deter, to make sure that users know that their actions are being watched and recorded.

With these wishes in mind, ICL decided to transform the Audit capabilities of the system under the HSO. Existing audit facilities were supplemented by the provision of a complete new set of security audit records with special message types and subject to special protection.

The following events can be audited under the new scheme:

- attempted security violations,
- logon, jobstart, session initiation and termination,
- submission of incorrect logon data,
- changes to mandatory and discretionary security policy,
- changes to audit policy,
- the exercising of nominated types of trust under the mandatory policy,
- procurement of printed output,
- loading of code from nominated libraries,
- changes to passwords, but not the values involved,
- changes to security label values,
- any access by any user to any protected VME object.

The last of these event categories has the potential to generate an unacceptably large amount of audit data, so a wide variety of subsetting options have been made available to the audit manager.

These are:

- accesses by nominated users,
- accesses using nominated workstations,
- accesses using nominated services,
- accesses when code from a nominated library is available for execution,
- read accesses to objects whose confidentiality exceeds a nominated threshold,
- write accesses to objects whose integrity exceeds a nominated threshold,
- accesses to particular nominated objects or object types.

Together, these features permit an Audit Manager to define the precise audit profile he requires for his system. This can be varied on a day to day basis to react to changing circumstances.

In designing the formats of the new audit records ICL had a choice to make between human readable but large and therefore inefficient formats, and compacted machine processable formats that are more difficult for a human to interpret directly. ICL opted for the latter on the basis that in the future the raw data from audit trails will increasingly require extensive machine pre-processing in order to provide statistical data, to highlight significant events and to analyse for unusual changes in user work patterns that may indicate potential attacks on the system (Ref 8). A demonstrator for analysing VME audit trails has already been developed by Logica in the UK under the auspices of the British Government's Central Computer and Communications Agency (CCTA).

8. OTHER FEATURES

Space does not permit a full description of all of the security features of the VME HSO, indeed it is ICL's policy to restrict the availability of full details of the product to only bonafide customers having a legitimate interest in it (for such customers a range of six security manuals has been produced). Other enhancements to the system's security have however been made in the areas of authentication, security labelling of printed output, and the protection of discarded data; this paper has only hinted at the full power of the "trusts" system.

A range of enforcement options has also been implemented to permit a customer to move gradually and painlessly into a secure mode of working following purchase and installation of the HSO.

9. WHAT IS THE CUSTOMER TO MAKE OF ALL THIS COMPLEXITY?

A question like this is understandable, and ICL has been very conscious that the power and potential complexity of the features provided by the HSO can be rather intimidating. The company has therefore provided a comprehensive training and consultancy support programme to supplement the HSO product itself. It should be remembered that it is very much in ICL's interest to make sure that the introduction of the leap forward in security that this product represents is a success.

It is also very important to note that a customer's philosophy when implementing his particular security policy should be "keep it simple and stupid!". Although the VME HSO provides a rich and complex variety of controls, they are there as a shopping list from which each different customer is free to select his own simple profile; the complexity is ICL's not the customer's. If a system uses only a small proportion of the features offered by the VME HSO, but as a result, sensitive and valuable information is protected to a high level of assurance, then use of the product has been worthwhile. A customer should not feel that in order to get value for money, all of the security features of the system must be exercised to their fullest extent, indeed such an approach would be likely to achieve just the opposite effect.

10. WHAT THE VME HSO DOES NOT DO

The VME HSO is an operating system development. It protects objects that are understood by the basic VME operating system. This means it protects things like whole files, libraries, disc partitions, communications devices and magnetic tape drives. It does not directly protect application-level objects like individual data items in an IDMS or Ingres database. To VME a database is a file, or at most a few files. The database is protected therefore at exactly that level of granularity and its individual data items are protected by the HSO only as a consequence of the protection afforded to the files that contain them.

Similarly, any software that runs on top of VME is treated by the VME HSO as being untrusted unless it is explicitly told otherwise by the security manager. This normally means all application packages and all superstructure products, including TPMS, are treated by the VME HSO as untrusted. This is not to say of course that these products are not worthy of any trust; indeed many installations will utilise their protection features to supplement the security provided by the VME HSO, whose features might then be looked upon as providing the secure environment within which individual applications can operate.

11. NEXT STEPS

At the time of writing of this paper, only a few customers have obtained experience of this new product. Choice of future enhancements will therefore depend very much on feedback which is yet to be obtained. There are however two areas of future development which are worth highlighting:

The first is in the area of FTAM, or " File Transfer and Access Method". An early implementation of this feature on VME will permit VME HSO customers to transmit a file's security label along with the file, and therefore allow such a file to be protected in its destination system in the same way as on its source system.

The second development is in the area of user authentication. In the distributed systems of the future, users will wish to authenticate themselves once to the system as a whole, and use the results of this to access all of the applications he wishes, in a standard manner, no matter which particular end systems contain them. ICL is developing such a network Authentication Server and the VME HSO will be adapted to accept the resulting certified identities rather than repeat the authentication process by engaging in VME logon exchanges. VME usernames will then become resources which individual users may or may not be permitted to access. By this means accountability of individual human users will be achieved no matter how many share the use of the same VME username.

12. CONCLUSIONS

The American evaluation scale and its UK equivalent represent a major step forward in our understanding of what is required of a secure computer system. Level B systems on these scales will give a significantly better level of security protection than the conventional level C systems of today. These benefits will not come without effort on behalf of both users in managing their systems securely, and manufacturers in giving them the technical tools with which to do this; the potential gain in our ability to protect data in computer systems is however enormous.

Of prime importance is the support of a Mandatory Confidentiality Policy, without which the higher level of security assurance provided by level B systems cannot be obtained. The commercial world requires similar assurances with respect to integrity, and it is possible to satisfy these requirements by providing support for a Mandatory Integrity Policy. In both cases however, the implementation needs to be rich and flexible; a simple implementation would have unacceptable usability consequences.

Access control is not by itself sufficient; users must be made accountable for their actions and an audit capability of similar power and flexibility is also required.

The VME High Security Option provides these and other related security facilities for ICL's customers. It has passed its first hurdle: successfully achieving British Government evaluation to the UK equivalent of B1 on the US DoD scale. The next hurdles may be even more difficult: proving its usability, manageability and security against real attack in real customer environments.

References

1. PARKER T.A.: "ICL Efforts in Computer Security", Proceedings of the 4th Seminar of the DoD Computer Security Initiative, August 10-12 1981.

2. DOD 5200.28-STD.: "Trusted Computer Systems Evaluation Criteria", Fort George Meade MD USA: National Computer Security Center, December 1985.

3. CESG: "UK Systems Confidence Levels", CESG Computer Security Memorandum No. 3, Issue 1.1, Feb. 1989.

4. GERMAN CIPHERBOARD: "National Catalog of Criteria for the Evaluation of Trusted IT Systems", Draft version, issued by the German Cipherboard, Federal Republic of Germany.

5. DTI: "Evaluation and Certification Manual", V23 - Version 3.0, one of five volumes produced by the DTI Commercial Computer Security Centre, RSRE Malvern, Worcs.

6. BELL D.E. and LAPADULA L.J.: "Secure Computer Systems: Unified Exposition and Multics Interpretation", MTR-2997 Rev.?1 MITRE Corp., Bedford, Mass. March 1976.

7. CLARK D.D. and WILSON D.R.: "A Comparison of Commercial and Military Computer Security Policies", in Proc. Symp. on Security and Privacy, IEEE, April 1987.

8. TERESA F. LUNT: "Automated Audit Trail Analysis and Intrusion Detection: A Survey", in Proc. 11th National Computer Security Conference, Baltimore, October 1988.

9. "Information Technology Security Evaluation Criteria (ITSEC)", Harmonised Criteria of France - Germany - The Netherlands - The United Kingdom, 2nd May 1990 Version 1 (Draft).

Rainbows and Arrows : How the Security Criteria Address Computer Misuse

Peter G. Neumann
Computer Science Lab, SRI International, Menlo Park CA 94025-3493
Neumann@csl.sri.com Phone 415-859-2375

13th National Computer Security Conference
Washington DC, 1-4 October 1990

Abstract

This paper examines the two main sets of computer security evaluation criteria and considers the extent to which each criterion combats various types of threats. Differences among the criteria sets are summarized, and recommendations are offered for improved coverage.

Introduction

In the 1989 National Computer Security Conference, Neumann and Parker [89] considered various classes of techniques for intentional or accidental misuse of computers and communications. Table 1 gives a terse summary of the misuse classes and illustrations of various types of misuse techniques. Exploitations frequently involve multiple techniques used in combination.

Two sets of security criteria are considered here, the U.S. Trusted Computer Security Evaluation Criteria (TCSEC) and the European Information Technology Security Evaluation Criteria (ITSEC). Each has certain strengths and certain deficiencies. Together they remain incomplete in their coverage and not completely consistent with one another. Nevertheless, they must be considered as part of an evolutionary process, and represent important steps toward improved system security.

Both criteria sets are threat oriented. They are themselves evaluated here with respect to the specific threats that they do or do not address.

The TCSEC

The Trusted Computer Security Evaluation Criteria (TCSEC) of the United States Department of Defense are summarized in Figure 1, which is reproduced from TCSEC [85] (the Orange Book). Apart from the degenerate D class, each evaluation class (designated C1, C2, B1, B2, B3, A1, in order of generally increasing functionality and assurance) has associated with it a collection of criteria that address security policy, accountability, assurance, and documentation. The criteria for any evaluation class subsume the criteria at lesser classes. Many of the criteria elements have different implications at different evaluation classes; for example, security testing appears in Figure 1 with increasingly stringent requirements at each evaluation class from C1 to A1. Overall, the C class corresponds to conventional threats, the B class to more severe threats, and the A1 class provides greater assurance for B3 functionality.

A distinction is made between the ratings of products and the security of installed systems. Actual configurations of systems, particularly when networked, may result in vulnerabilities in spite of the evaluated ratings of individual component products. For example, passwords transmitted in the clear between networked systems may permit easy system compromise. Similarly, a flawed *sendmail* can undermine systems through dial-up lines, even without networking. Consequently, it is vital to consider each system complex (including networks, distributed system control, database management, and applications) as a single system. For this purpose, the Trusted Network Interpretation (TNI, Red Book, TCSEC-TNI [87]) and the Trusted Database Interpretation (TDI, TCSEC-TDI [89]) should be considered in addition to the Orange Book, along with others in the 'rainbow' series of documents -- which help to put TCSEC [85] in context. Analysis of a composite system may benefit from component evaluations; however, because the TCSEC were established before composite systems had become better understood, there are some basic shortcomings.

The Harmonised ITSEC

The Information Technology Security Evaluation Criteria (ITSEC), the Harmonised Criteria of France, Germany, the Netherlands, and the United Kingdom (ITSEC [90]) represent an effort to establish a comprehensive set of security requirements for widespread international use. ITSEC is generally intended as a superset of TCSEC, with ITSEC ratings mappable onto the TCSEC evaluation classes (see below). Historically, ITSEC represents a remarkably facile evolutionary grafting together of the evaluation classes of the German [light] Green Book ('das grüne Buch', GISA [89]) and the 'claims language' of the British [dark] Green Books (DTI [89]). (The predecessor criteria are considered here only in passing. Brunnstein and Fischer-Huebner [90] and Pfleeger [90] contrast these criteria with TCSEC.)

- **EX:** External abuse
    1. Visual spying: observation of keystrokes or screens
    2. Misrepresentation: deception of operators and users
    3. Physical scavenging: dumpster-diving for printout
- **HW:** Hardware abuse
    4. Logical Scavenging: examining discarded/stolen media
    5. Eavesdropping: electronic or other data interception
    6. Interference: electronic or other jamming
    7. Physical attack on or modification of equipment or power
    8. Physical removal of equipment and storage media
- **MQ:** Masquerading
    9. Impersonation (false identity external to computer systems)
    10. Piggybacking attacks (on communication lines, workstations)
    11. Playback and spoofing attacks
    12. Network weaving to mask physical whereabouts or routing
- **PP:** 'Pest' programs (setting up further abuses)
    13. Trojan-horse attacks (including letter bombs)
    14. Logic bombs (including time bombs), a form of Trojan horse
    15. Malevolent worm attacks, acquiring distributed resources
    16. Virus attacks, attaching to programs and replicating
- **BY:** Bypassing authentication/authority
    17. Trapdoor attacks (due to any of a variety of sources) --
        - a. Improper identification and authentication
        - b. Improper initialization or allocation
        - c. Improper termination or deallocation
        - d. Improper validation
        - e. Naming flaws, confusions, and aliases
        - f. Improper encapsulation: exposed implementation detail
        - g. Asynchronous flaws: time-of-check to time-of-use anomalies
        - h. Other logic errors
    18. Authorization attacks (e.g., password cracking, token hacking)
- **AM:** Active misuse of authority (writing, using, with apparent authorization) --
    19. Creation, modification, use (including false data entry)
    20. Incremental attacks (e.g., salami attacks)
    21. Denials of service (including saturation attacks)
- **PM:** Passive misuse of authority (reading, with apparent authorization) --
    22. Browsing randomly or searching for particular characteristics
    23. Inference and aggregation (especially in databases), traffic analysis
    24. Covert channel exploitation and other data leakage
- **IM:** 25. Misuse through inaction: willful neglect, errors of omission
- **IN:** 26. Use as an indirect aid for subsequent abuse: off-line preencryptive matching, factoring large numbers, autodialer scanning.

Table 1: Summary of Techniques for Computer Misuse

	SECURITY POLICY											ACCOUNTABILITY			ASSURANCE									DOCUMENTATION			
	DISCRETIONARY ACCESS CONTROL	OBJECT REUSE	LABELS	LABEL INTEGRITY	EXPORTATION OF LABELED INFORMATION	EXPORTATION TO MULTILEVEL DEVICES	EXPORTATION TO SINGLE-LEVEL DEVICES	LABELING HUMAN-READABLE OUTPUT	MANDATORY ACCESS CONTROL	SUBJECT SENSITIVITY LABELS	DEVICE LABELS	IDENTIFICATION AND AUTHENTICATION	AUDIT	TRUSTED PATH	SYSTEM ARCHITECTURE	SYSTEM INTEGRITY	SECURITY TESTING	DESIGN SPECIFICATION AND VERIFICATION	COVERT CHANNEL ANALYSIS	TRUSTED FACILITY MANAGEMENT	CONFIGURATION MANAGEMENT	TRUSTED RECOVERY	TRUSTED DISTRIBUTION	SECURITY FEATURES USER'S GUIDE	TRUSTED FACILITY MANUAL	TEST DOCUMENTATION	DESIGN DOCUMENTATION
A1	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	▨	▨	▨	□	▨	□	▨	□	□	▨	▨
B3	▨	□	□	□	□	□	□	□	□	□	□	□	▨	▨	▨	□	▨	▨	▨	▨	□	▨	■	□	▨	□	▨
B2	□	□	▨	□	□	□	□	□	▨	▨	▨	□	▨	▨	▨	□	▨	▨	▨	▨	▨	■	■	□	▨	▨	▨
B1	□	□	▨	▨	▨	▨	▨	▨	▨	■	■	▨	▨	■	▨	□	▨	▨	■	■	■	■	■	□	▨	□	▨
C2	▨	▨	■	■	■	■	■	■	■	■	■	▨	▨	■	▨	□	▨	■	■	■	■	■	■	□	▨	□	□
C1	▨	■	■	■	■	■	■	■	■	■	■	▨	■	■	▨	▨	▨	■	■	■	■	■	■	▨	▨	▨	▨

□ NO ADDITIONAL REQUIREMENTS FOR THIS CLASS

▨ NEW OR ENHANCED REQUIREMENTS FOR THIS CLASS

■ NO REQUIREMENTS FOR THIS CLASS

Figure 1: TCSEC Summary Chart

ITSEC unbundles *functional criteria* (F1 to F10) and *correctness criteria* (E0 as the degenerate case, and E1 to E6), which are evaluated independently.

The functional criteria F1 to F5 are of generally increasing merit, and correspond roughly to the functionality of C1, C2, B1, B2, and B3, respectively. The remaining functionality criteria address data and program integrity (F6), system availability (F7), data integrity in communication (F8), data confidentiality in communication (F9), and network security including confidentiality and integrity (F10). F6 to F10 may in principle be evaluated orthogonally to each other and to the chosen base level, F1, F2, F3, F4, or F5.

The correctness criteria are intended to provide increased assurance. To a first approximation, the correctness criteria cumulatively require testing (E1), configuration control and controlled distribution (E2), access to the detailed design and source code (E3), rigorous vulnerability analysis (E4), demonstrable correspondence between detailed design and source code (E5), and formal models, formal descriptions, and formal correspondences between them (E6). E2 through E6 correspond roughly to the assurance aspects of C2, B1, B2, B3, and A1, respectively.

An ITSEC rating is thus *one or none* of F1 to F5, *one* of E0 to E6, and *one or none of each* of F6 to F10, i.e., one of 6x7x2x2x2x2x2 = 1344 ratings. The intended approximate mappings from ITSEC functionality and correctness to TCSEC evaluation classes are given in Table 2 (although the respective definitions are not always completely consistent). F6 to F10 do not enter into the mapping, as they have no direct correspondence in TCSEC.

ITSEC function level	ITSEC correctness level	TCSEC evaluation class
	E0	D
F1	E2	C1
F2	E2	C2
F3	E3	B1
F4	E4	B2
F5	E5	B3
F5	E6	A1

Table 2: Mapping of ITSEC *onto* TCSEC

Because of the unbundling of functionality and assurance, other combinations such as F4/E3 are potentially meaningful. However, extreme combinations such as F5+6+7+8+9/E0 and F1/E6 are unrealistic. In any event, the mapping from ITSEC to TCSEC is many-to-one (e.g., F4/E3 and F3+7/E3 both map to B1), and therefore not uniquely reversible in the absence of the original ITSEC context (i.e., B1 maps back to F3/E3).

ITSEC's unbundling has advantages and disadvantages. On the whole it is a meritorious concept, as long as assurance does not become a victim of commercial expediency, and if the plethora of rating combinations does not cause confusion.

Although ITSEC [90] contains nothing analogous to Figure 1, there is a comparable table in the precursor German criteria document (GISA [89], pp. 106-107) for its functional and 'quality' (now 'correctness') criteria, distinguishing as in Figure 1 between 'no requirement', 'new requirement', and 'no new requirement'. Because Figure 1 is so useful as a definitional reference, a similar table would be useful for ITSEC.

ITSEC addresses "generic headings" of identification and authentication, access control, accountability, audit, object reuse, accuracy, reliability of service, and data exchange. A semiformal *claims language* is used to define particular properties that must be satisfied. The claims provide the basis for evaluation or self-evaluation. Table B.1 of ITSEC [90] shows the relationships between 35 claims-language "target phrases" and the generic headings.

The Criteria and the Misuse Techniques

This section contrasts TCSEC and ITSEC, and also discusses their applicability to the various misuse techniques. Table 3 indicates which misuse techniques (Table 1) are addressed by each of the two sets of evaluation criteria, TCSEC and ITSEC. The technique-type numbers and symbolic class designators in Table 3 are those noted in Table 1. An entry in the body of Table 3 implies that the particular criteria element contributes something constructive to the prevention or detection of the indicated misuse technique or class. However, because of the inherently weak-link nature of security, it is necessary to consider the coverage provided by the totality of all criteria rather than that of any individual criterion.

The first section of Table 3 summarizes the misuse techniques relative to the TCSEC evaluation criteria (Figure 1). Apart from questions of the extent of protection and assurance, the TCSEC entries of Table 3 are relatively independent of the specific evaluation classes, for those evaluation classes for which requirements exist (i.e., for which the matrix entry in Figure 1 is not black).

To the extent that the ITSEC criteria F1 to F5 map onto the TCSEC criteria (when combined with the correctness criteria, as noted in Table 2), the ITSEC functionality classes F1 to F5 can be related directly to the first section of Table 3 via the particular combination of TCSEC requirements in Figure 1. The additional functionality criteria (F6 to F10) are only partially covered by TCSEC and TNI. The relevance of ITSEC to the misuse techniques is summarized in the second section of Table 3.

Criterion \ Misuse Techniques	External	Hardware	Masquerading	Pests, Bypasses	Active Misuse	Browsing	Inference	Covert channels	Inaction	Indirect aid
	EX 1-3	HW 4-8	MQ 9-12	PP,BY 13-18	AM 19-21	PM 22	PM 23	PM 24	IM 25	IN 26
TCSEC Security Policy:										
Discretionary access control				(*)	*	*				
Object reuse				17b,c						
Labels & label integrity				*	*	*	*	*		
Exportation (3 criteria)		(4)		*	*	*	*	*		
Labeling human-read output	(3)			*	*	*	*	*		
Mandatory access controls				*	*	*	*	*		
Subject sensitivity labels				*	*	*	*	*		
Device labels				*	*	*	*	*		
TCSEC Accountability:										
Identification/authentication			(9)	*	*	*	(*)	(*)		
Audit			(*)	*	*	*	*	*	(*)	(*)
Trusted path			*	*						
TCSEC Assurance:										
System architecture			(*)	*	*	*	*	*		
System integrity				*						
Security testing			(*)	(*)						
Design spec/verification				*	*	*	(*)	*		
Covert channel analysis						(*)	(*)	*		
Trusted facility management	(3)			(*)	*	*	(*)			
Configuration management			(*)	*						
Trusted recovery			(*)	*						
Trusted distribution				*						
TCSEC Documentation:										
Security features user's guide				*	*	*	*	*		
Trusted facility manual				*	*	*	*	*		
Test documentation				*	*	*	*	*		
Design documentation				*	*	*	*	*		
ITSEC Functionality and Correctness:										
F1. C1 functionality				(*)	*	(*)				
F2. C2 functionality				(*)	*	(*)				
F3. B1 functionality	(3)	(4)		(*)	*	(*)				
F4. B2 functionality	(3)	(4)	(*)	*	*	*	*	*		
F5. B3 functionality	(3)	(4)	(*)	*	*	*	*	*	(*)	(*)
F6. Data/program integrity				*	*					
F7. System availability		(6-8)		*	*	(*)				
F8. Comm data integrity		6		(*)	*					
F9. Comm data confidentiality		5		(*)	*	*	*	(*)		
F10.Network security/integrity		5	*	*	*		(*)	(*)		
E1-E3, with varying assurance				(*)	(*)	(*)	(*)			
E4-E6, with varying assurance				(*)	(*)	(*)	(*)	*		

Legend: The column-head misuse class designators and the misuse-type numbers refer to those in Table 1.
Misuse types are grouped according to similar characteristics.
'*' implies the given criterion helps generally to combat the misuse class(es) in the column head.
Numbers imply only certain misuse types are applicable within the column-head class(es).
Parentheses imply a secondary effect for the particular criterion and misuse class(es) or type.
Refer to Figure 1 for relevant TCSEC evaluation classes for each TCSEC criterion.

Table 3: Criteria relevance for combatting misuse techniques

To the extent that ITSEC is a proper superset of TCSEC, many of the following comments about TCSEC are also relevant to ITSEC. When ITSEC is discussed *per se*, it is usually where it differs from TCSEC.

TCSEC bundles its criteria in two dimensions, as can be seen in Figure 1. First, functionality and assurance criteria are coupled rather rigidly. Second, each evaluation class is considered as a monolithic collection of criteria; in practice it would be useful to define intermediate evaluation classes such as 'C2+' or 'B1+", defined with certain specified features of higher classes and extra requirements (e.g., akin to F6 to F10).

The TCSEC criteria do not adequately address availability, data integrity (such as assurances that files have not been tampered with through bypasses to the write-protection mechanism), and generalized nondenial of service, for example. (The ITSEC criteria are somewhat more explicit about requiring availability and preventing denials of service.) The TCSEC criteria also do not address trusted paths to and authentication by virtual systems that do not have comparable facilities with respect to the end users, although extensions have been proposed. Furthermore, there is still some uncertainty about the criteria-relevant effects of layered trusted computing bases (TCBs). These considerations, together with the proliferation of TCSEC 'interpretations' (e.g., TNI and TDI for networks and databases, respectively), indicate that there are additions to TCSEC that would be relevant; indeed, the ITSEC F6-F10 have attempted to address some of them. All of the misuse techniques of Table 1 are relevant to distributed systems, networks of computer systems, and database systems, and thus need to be covered explicitly by any subsequent extensions or modifications to the criteria.

The ITSEC F1-F5 functional criteria (together with the appropriate correctness criteria) map fairly well onto the TCSEC requirements, according to Table 2, while F6-F10 do not. For F6 to F10, it is unclear what correctness criteria would be meaningful in isolation, particularly because failure to enforce the F6-F10 requirements with adequate assurance could actually undermine the enforcement of overall system security supposedly covered by the F1 to F5 rating. For example, inadequate attention to integrity, communications, or networking can undermine the security of installed computer systems. The *sendmail debug option* problem provides an illustration.

Defensive measures should be chosen to prevent wasteful coverage of nonthreats and to prevent gaps from existing at the interfaces among the various measures. Indeed, the 'Chinese Menu' flavor of the ITSEC criteria (i.e., the unbundling of functionality and correctness, plus the ostensibly orthogonal F6 to F10 requirements) appears to be attractive for that reason. However, many of 1344 potential ratings of ITSEC functionality and correctness are not particularly logical, consistent, or sound, and should be avoided; in contrast, the mapping (Table 2) of ITSEC ratings onto one of only seven TCSEC evaluation classes suggests that TCSEC might be too monolithic.

Further Discussion

Security requires an overall systems view, and all potential weak links must be considered. TCSEC and ITSEC focus on certain basic aspects of system misuse, but are less comprehensive in others. In this section we consider the roles of security policy, accountability, and assurance, as well as the special problems of networks and databases.

Security policy

Table 3 suggests that security policy criteria help to address the basic misuse types (13-24), but the table does not indicate the extent to which weak-link phenomena predominate. In particular, examination of the column for pest programs and bypasses shows that the problems of preventing these forms of attack are rather pervasive, in that *every one* of the criteria elements contributes something to combatting these attacks, but that in combination all of the criteria are still not quite enough. Penetrators typically appear as if they are authorized users. Pest programs are especially insidious because they execute on behalf of authorized users, with the normal privileges of their unsuspecting victims. Although any particular *known* personal computer virus (or propagating Trojan horse) that does not mutate may be detectable, viruses are *in general* very difficult to detect -- especially if they resort to techniques such as mutation, length-preserving compressions, and dispersion into small pieces. Such techniques escalate pest-program defense to being 'beyond feasibility' in general.

Thus, a combination of all of the cited criteria elements (including better PC hardware and operating systems) evidently would help somewhat, but would still not be enough. Finer-grain access controls that closely reduce what is permitted to just what is actually necessary can help to combat these attacks, including misuse by apparently authorized users, by narrowing the basic gap that otherwise prevents access controls from enforcing what is actually intended.

The existence of compartmented multilevel security (MLS) tends to limit some of the adverse effects from pest programs and bypasses -- notably adverse flow of information -- as well as reducing opportunities for misuse by authorized users. MLS is of potential value throughout a distributed system or network, assuming that there is comparable trustworthiness in enforcement. Some sort of mandatory integrity (e.g., the restrictive multilevel integrity, MLI, of Biba [75], or the more flexible type-based integrity provided by LOCK, Boebert [85]) can also help, particularly in preventing trusted applications from depending on less trusted programs and data, assuming explicit or implicit certification of new programs and data. Denials of service could be restricted by the combination of MLS and MLI, at least by confining the effects within security/integrity levels and compartments. However, even with such multilevel controls there are still vulnerabilities, such as malicious deletion within the same level and compartment. The application integrity policy of Clark and Wilson [87] also provides a significant set of criteria, relating to good software engineering practice.

The scope of coverage is quite diverse for the various security-policy related criteria elements. One of the more narrowly defined requirements is the TCSEC criterion for proper object reuse, addressing improper initialization or allocation (type 17b) and also relating to improper termination or deallocation (type 17c) in Table 3. Its proper enforcement depends on noncompromisability of other criteria. For the general technique class of bypassing authentication and authority (BY) in Table 3, preventing the subtypes of trapdoor attacks (type 17) requires intelligent software development; object reuse is just one specific example of this need. Thus, implicit in the process of adhering to the criteria is a requirement that demands better system engineering, including software, hardware, and the operating environment. Also, inherently weak security policies (e.g., C2 discretionary access) should not be relied on in critical applications.

Accountability

Passwords provide a fundamentally flawed authentication mechanism, although neither criteria set adequately reflects the seriousness of the problems. For example, there is a B1 requirement for authentication, but nothing higher except for the trusted path requirements at B2 and B3, which only slightly reduce the threats to password compromise. Something more stringent (such as encryption-based authenticators) is undoubtably desirable in sensitive environments, although even those mechanisms are vulnerable to certain forms of compromise.

Logging and auditing play a vital role throughout. (Auditing is the only criterion that addresses the rather obscure techniques of misuse through inaction (type 25) and use as an indirect aid (type 26), and then only *after the fact*.) Although not addressed in detail in either of the criteria sets, real-time audit-trail analysis is expected to become a major contributor in the future, in hopes of catching perpetrators *in flagrante delicto*. Lunt [88] surveys real-time analysis systems that use rule-based expert systems and/or profile-based statistical systems. Real-time analysis has the potential of providing additional deterrents that *post-hoc* analysis cannot.

Nonrepudiation is a rather specific requirement (e.g., DTI [89]), addressing a small corner of the authentication problem in which authenticity cannot easily be denied at a later time, i.e., part of the attack technique of improper identification and authentication (type 17a). Nonrepudiation was present in the predecessor Dark Green books, but appears only implicitly in ITSEC.

Assurance

Examination of Table 3 indicates that the criteria only incidentally address external abuse and hardware abuse (technique types 1 to 8). Protection against emanations (part of type 5) and interference (type 6) is extensively covered elsewhere for military and intelligence applications, but is widely ignored in other applications. Nevertheless, stray emanations and interference have been responsible for human deaths in life-critical applications (e.g., the combination of microwave emanations and heart-pacemaker interference), and must be recognized as both security and integrity problems in critical environments. More generally, better administrative guidance for external and hardware abuse would be appropriate, particularly for unclassified critical applications.

Physical security is an important part of defending against various classes of attack; it is generally thought of in relation to hardware abuse and certain external attacks, but often is relied upon implicitly for authentication, trusted path, and configuration management criteria as well. It is usually considered separate from computer security, less glamorous in its nonresearch nature. However, physical access to computer and communication equipment can seriously undermine the ability to enforce TCB security and integrity. For example, the trusted distribution requirement (relating to some assurances that the system is untampered with) arises in TCSEC only at A1, but is generally relevant; trusted paths arise at B2 and B3, but are also meaningful below that. Defending against the insertion of pest programs and trap doors depends not only on the cited criteria but also to some extent on physical security and people, especially in personal computers.

Operational security is another serious concern. Trusted facility management has requirements at the B2 and B3 classes relating to the separation of duties between operator and administrator roles (B2) and additionally between security administrator and system administrator roles (B3). (Separation of duties more generally is fundamental to the Clark and Wilson integrity model, and is not explicitly addressed by either of the criteria sets.)

Databases

For database management systems, all of the basic misuse techniques and all of the criteria elements of Table 3 are relevant. Of particular importance are database issues relating to integrity, inference, and covert channels, at a granularity different from operating systems:

- Integrity constraints often lead to confusion in databases. Consistency of distributed and/or replicated data has both security and integrity implications. Primary-key uniqueness and referential integrity have both integrity and inference implications. Integrity locks (e.g., cryptological seals) are of interest, although compromises via the underlying operating system must be considered.

• Inference issues are intrinsic in databases, and generally impossible to combat completely.

• Covert channels arise for a variety of reasons, including the use of shared indices, concurrency controls, resource exhaustion, recovery, shared devices, and naming conflicts. They are also a common side-effect of discretionary access control mechanisms. They are seemingly more difficult to control in databases than in operating systems, especially when data dependent.

The TCSEC Trusted Database Interpretation (TCSEC-TDI [89]) gives considerable guidance on such issues. Databases are of interest to the ITSEC criteria only as instances of entire systems. Issues of hierarchically layered assurance raised by the TDI are in the long run likely to be very important, whenever systems are composed out of components with different degrees of trustworthiness. The absence of explicit layering of TCBs in the ITSEC criteria suggests that the entire DBMS and underlying operating system might have to be evaluated as one, rather than being able to reason about the underlying TCB. Nevertheless, compositional reasoning is plausible within ITSEC. (Discussion of balanced assurance versus uniform assurance must await the final version of the TDI.)

As an example of a database system targeted for a TCSEC B3 or A1 rating, the SeaView security/integrity model (Denning et al. [88]) and system design (Lunt et al. [88]) provide a general approach capable of advanced database security, including multilevel security. The SeaView architecture involves layers of trustworthiness, based on a multilevel secure trusted computing base (Gemini's GEMSOS), and a slightly modified commercial DBMS (Oracle). The database engine is untrusted for multilevel security, but is trusted for integrity. SeaView explicitly addresses a wide range of security and integrity threats (including misuse techniques 13 through 24).

Networks

The TNI and ITSEC F8, F9, and F10 are particularly relevant to networks; essentially all of the misuse techniques are applicable to computer-communication networks *per se* (irrespective of the computer systems that they conjoin), although the coverage in Table 3 is somewhat spotty. For example, the MILNET terminal access concentrators (TACs) provide dial-up or hard-wired access to all systems on MILNET. The TACs and the interface message processors (IMPs) are logically internal to the network, and invisible to ordinary programmers. Because the TACs and IMPs are systems (nodes) without 'users', some of the techniques may at first seem less applicable, such as the ability of unauthorized people to insert pest programs. However, such vulnerabilities still exist, because of the way in which program maintenance is done remotely using the network itself. This suggests that without careful consideration it is dangerous to assume that any of the criteria elements is *not* applicable. The 27 Oct 1980 Arpanet collapse and the 15 Jan 1990 AT&T slowdown both indicate how a system flaw can accidentally result in the propagation of damage throughout the network. (See Neumann [90a].) There is also an important security lesson to be learned from such accidental problems, because both could alternatively have been triggered intentionally. Thus, networking appears to require still broader coverage of vulnerabilities.

Other Criteria

The Canadian draft criteria (CSE [89]) outline classes A, B, C, and D, as in TCSEC, as well as divisions of integrity (E,F,G,H), availability (J,K,L,M), accountability (P,Q,R,S), and trustworthiness (T0, T1, T2, etc.). Draft French criteria also exist, in the "Blue-White-Red Book". (Harmonization of those two criteria sets could result in colorful gourmet alphabet soup. Vive la différence!)

The British Ministry of Defence has established a different set of standards (MoD [89]) for safety-critical computer systems. Those criteria require significant use of 'semiformal' methods. Indeed many of the requirements for secure systems are also relevant for life-critical systems, but by no means sufficient.

Conclusions

Table 3 represents an oversimplified effort to capture the essence of the relationships between the two criteria sets and the threats they seek to address. The reality is obviously more multidimensional, with some subtle distinctions among the different evaluation classes and among the different technique types within each misuse class. Nevertheless, the intent of this paper is to educe the major issues for deeper examination.

The two criteria sets have tended to focus to date largely on the simplest threats in relatively homogeneous systems. However, as technology and assurance measures both improve, as distributed sytems become more widespread, and as sophistication on the part of misusers increases, the serious threats may tend to change in nature and escalate in technology. Thus, it is important to anticipate such trends and ensure that the criteria cover all of the realistic threats. In general, this may result in a slow migration to intermediate or even higher functionality and assurance (whether currently defined or not), even in personal computers and workstations, and with particular attention to distributed systems and networking.

TCSEC and ITSEC are seen here to play useful roles in combatting malicious (and to some extent unintentional) misuse of computer systems and networks. However, both criteria sets reflect some vestiges of their historical perspective (despite the recency of ITSEC); important classes of misuse and various advanced architectures are not adequately covered. In addition, there are many related issues that are shortshrifted, such as more explicit recognition of the software-engineering relevance of the application integrity policy of Clark and Wilson, the importance of reusability and composability of sound building blocks such as TCSEC TCBs, and the fundamental nature of authentication. Trustworthy identification and authentication are vital to distributed systems. Also important are generality and flexibility in evaluation of real systems, e.g., evaluating system products and modifications generically, while also evaluating specific installations in their live environments. Neither of the two criteria sets deals satisfactorily with the assurance that results from hooking together either homogeneous or heterogeneous system components, reflecting the vulnerabilities in layered, networked, and distributed systems, although the Trusted Network Interpretation (TNI) and ITSEC F8-F10 attempt to address these issues. Some major remaining research issues were exposed in the TDI attempts to properly address layering of trusted components, and must be resolved.

Recommended Extensions

Both sets of criteria represent significant efforts to improve security in general, and to reduce the risks of malicious misuse in particular. Several specific recommendations for desirable extensions are noted below.

- Further system integrity is desirable above C1, e.g., to hinder system tampering.

- Mandatory integrity mechanisms can reduce the dependence on untrustworthy code and data.

- Application integrity *a la* Clark-Wilson can limit malicious code and other problems in applications.

- Availability measures (including the use of integrity requirements) can limit denials of service by both unauthorized and authorized users.

- Higher-assurance authentication is desirable above B1. Passwords generally have too many vulnerabilities.

- Trusted distribution is desirable below A1; trusted recovery is desirable below B3; trusted paths may be relevant below B2. All three of these can have significant roles in the prevention of pest programs, even in C2 systems.

- Real-time audit-trail analysis has the potential to detect pest program hatching, penetrations, and misuses of authority preliminary to or concurrent with misuse.

- Greater attention to distinctions between products and operational systems is desirable, including more administrative and management guidelines, as well as the ability to accommodate compositions of evaluated products, installed systems, and incremental changes. Further guidance on how the criteria might help *installed systems* to prevent misuse would be very valuable.

- Composite systems must be addressed systematically.

- More attention should be given to formal code verification and other means of demonstrating whether code is consistent with its specifications. This is important in certain critical applications, not just for security but also where human safety and very high availability are vital. It is interesting that the precursor German criteria (GISA [89]) included a quality criterion Q7 (equivalent in spirit to the first incarnation of TCSEC's 'beyond A1'), which failed to harmonize into an ITSEC E7 ('beyond E6').

- Specific products and installed systems need to be better matched with the threats they are intended to address, addressing risks and cost benefits. Table 3 must be recognized as a superficial first step.

- ITSEC provides many challenges for an evolutionary next-generation TCSEC addressing the above points, and especially the issue of TCSEC/ITSEC reciprocity. However, it will be important not to introduce circular dependencies or inconsistencies between the two.

The original version of this paper contained the following comment, in light of the rainbow-colored criteria books: ''Although the number of still unused colors is rapidly dwindling, it is hoped that neither the intersection of the requirements nor the union of the colors red and green will be used, for that would result in a *little black book* for security.'' It appears that the considerable harmonization already achieved in the past year by ITSEC has significantly reduced that concern.

Various ''CLEFs'' (CESG-Licensed Evaluation Facilities) are being formed to carry out ITSEC evaluations. Recognizing the considerable advantages that can result from relatively unrestricted reciprocity, it is hoped that harmonization of U.S., U.K., and German interests (among others) can lead to accord and creative counterpoint among the at-least-treble CLEFs, particularly in staving off nationalistic self-interest.

Malicious misuse of computer systems can never be prevented completely, particularly when perpetrated by authorized users. Nevertheless, there are considerable benefits that can be gained from evaluations with respect to the criteria addressed above -- with the recognition that some threats are not covered in adequate detail. (Note that too much specificity is also a bad idea if it stifles design diversity and exacerbates different vendors' compatibility concerns.) Further work is urgently needed to refine and extend TCSEC and ITSEC into a unified, coherent, international, mutually useful, and modern set of criteria that more precisely address the vulnerabilities and threats to be avoided, including those in heterogeneous distributed systems. Rapid convergence on such a universal set of criteria will be essential to the development of appropriate future products, systems, and their evaluations. TCSEC and ITSEC must not be considered rigidly as gospel (or as competitors), and must rapidly evolve together into a unified whole. It will be important in the future to incorporate security, integrity, availability, guaranteed performance, safety (cf. MoD [89]), and other vital requirements within a common composite-system framework (Neumann [90b]), and to be able to enforce whichever of those requirements ·are necessary, so that there will be greater assurance that critical systems can simultaneously satisfy the combined set of requirements. (For example, see Neumann [86]). However, we must never assume perfection on the part of the computer systems and their user communities, and must design and use the technology accordingly.

Acknowledgements

The author is indebted to Teresa Lunt and Marjory Blumenthal for their helpful suggestions. This paper was prepared with support from National Science Foundation Grant CCR-8715419.

References

K.J. Biba [75], Integrity Considerations for Secure Computer Systems. Report MTR 3153, MITRE Corp., Bedford, Massachusetts, June 1975.

E. Boebert [85], A Practical Alternative to Hierarchical Integrity Policies. *Proc. Eighth National Computer Security Conference*, 30 September 1985.

K. Brunnstein and S. Fischer-Huebner [90], Analysis of "Trusted Computer Systems", *Proc. 6th International Conference on Information Security: SEC'90*, IFIP TC-11, Helsinki-Espoo, 23-25 May 1990.

D. Clark and D. Wilson [87], A Comparison of Commercial and Military Computer Security Policies. *Proc. 1987 IEEE Symposium on Security and Privacy*, Oakland, California, April 1987, pp. 184-194.

CSE [89], Communications Security Establishment, Canadian Trusted Computer Product Evaluation Criteria. Draft, version 1.0, May 1989.

DTI [89], Commercial Computer Security Centre, Department of Trade and Industry, volumes V01 (Overview Manual), V02 (Glossary), V03 (Index), V11 (Users' Code of Practice), V21 (Security Functionality Manual), V22 (Evaluation Levels Manual), V23 (Evaluation and Certification Manual), V31 (Vendors' Code of Practice), Version 3.0, February 1989.

D.E. Denning, T.F. Lunt, R.R. Schell, W.R. Shockley, M. Heckman [88], The SeaView Security Model. *Proc. 1988 IEEE Symposium on Security and Privacy*, April 1988, pp. 218-233.

GISA [89], IT-Security Criteria, Criteria for the Evaluation of Trustworthiness of Information Technology (IT) Systems. German Information Security Agency (ZSI), 11 January 1990.

ITSEC [90], Information Technology Security Evaluation Criteria, Harmonised Criteria of France, Germany, the Netherlands, and the United Kingdom. Draft Version 1, 2 May 1990. Available from UK CLEF, CESG Room 2/0805, Fiddlers Green Lane, Cheltenham U.K. GLOS GL52 5AJ, or ZSI, Am Nippenkreuz 19, D 5300 Bonn 2, West Germany.

T.F. Lunt [88], Automated Audit Trail Analysis and Intrusion Detection: A Survey. *11th National Computer Security Conference*, Baltimore, Maryland, 1988.

T.F. Lunt, R.R. Schell, W.R. Shockley, M. Heckman, D. Warren [88], A Near-Term Design for the SeaView Multilevel Database System. *Proc. 1988 IEEE Symposium on Security and Privacy*, April 1988, pp. 234-244.

MoD [89], Requirements for the procurement of safety critical software in defence equipment. Interim Defence Standard 00-55, Ministry of Defence, Directorate of Standardization, Kentigern House, 65 Brown St., Glasgow G2 8EX Scotland, U.K., May 1989.

P.G. Neumann [86], On Hierarchical Design of Computer Systems for Critical Applications. *IEEE Trans. Software Engineering* SE-12 9, September 1986, pp. 905-920.

P.G. Neumann [90a], The Computer-Related Risk of the Year: Distributed Control. *Proc. 5th COMPASS (IEEE)*, June 1990.

P.G. Neumann [90b], Towards Standards and Criteria for Critical Computer Systems. *Proc. 5th COMPASS (IEEE)*, June 1990.

P.G. Neumann and D.B. Parker [89], A Summary of Computer Misuse Techniques. *Proceedings of the 12th National Computer Security Conference*, Baltimore MD, 10-13 October 1989, pp. 396-407.

C.P. Pfleeger [90], Comparison of Trusted Systems Evaluation Criteria of the U.S., Germany, and Britain, *Proc. 5th COMPASS (IEEE)*, June 1990; based on earlier TIS Report #309, Trusted Information Systems, Inc., PO Box 45, Glenwood MD 21738, 2 March 1990.

TCSEC [85], Department of Defense Trusted Computer System Evaluation Criteria. DOD 5200.28-STD, December 1985 (Orange Book).

TCSEC-TDI [89], Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria. Draft, 25 October 1989, National Computer Security Center. (Revision pending.)

TCSEC-TNI [87], Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria. NCSC-TG-005 Version-1, 31 July 1987 (Red Book). (Revision pending.)

CIVIL AND MILITARY APPLICATION OF TRUSTED SYSTEMS CRITERIA

William C. Barker
Charles P. Pfleeger

Trusted Information Systems, Inc.
3060 Washington Road (Route 97)
Glenwood, MD 21738

Abstract

Trusted computer systems are commonly advertised in the context of military security requirements. Systems for military applications are designed to control access by need to know, by compartments, and by hierarchical levels. By introducing and defining a set of modes of operation for computer systems, then providing minimum evaluation criteria for systems operating in each mode, *Department of Defense (DoD)* guidelines can also be useful for civilian applications. To date, most development of trusted systems has been in response to military requirements, and the terminology and perceived usefulness of trusted systems development has tended to reflect this origin. It is, however, straightforward to map civilian needs—both functionality and assurance—onto military trusted systems.

Introduction

Many managers of non-military computer systems who perceive a need for computer security are frustrated by the scarcity of guidelines for selecting and applying trusted systems outside the defense and intelligence communities. The U.S. *DoD Trusted Computer Systems Evaluation Criteria [TCSEC]* is the only generally accepted criteria for evaluating the trustworthiness of computer systems in the United States. Trusted computer systems are commonly advertised in the context of military security requirements. Degrees of trustworthiness are expressed as digraphs (e.g., C2, B2, A1). Systems with lower ratings (e.g., C2) are designed to prevent the access to information by persons not having a legitimate "need to know" for that information. Systems granted higher ratings (i.e., B1) are designed to prevent the access to "compartmented" information by users not briefed into the "compartment," while even higher rated systems (i.e., B2, B3, and A1) are designed to prevent access to classified information by users not possessing security clearance for that information. However, there is little guidance concerning relevance of the *TCSEC* and attendant applications guidelines [CSC003] to civil requirements.

The DoD *Security Requirements for Automated Information Systems* [DoD28], while more overtly military in its audience, may provide a better foundation for determining trusted systems requirements than the National Computer Security Center's applications guidelines. By introducing and defining a set of modes of operation for computer systems, then providing minimum TCSEC ratings for systems operating in each mode, the DoD security requirements provide guidelines that are also useful for civilian applications. This paper is a brief description of different security attributes that may be associated with computer systems and the effects of those attributes on the modes in which the systems may safely operate. It attempts to treat both civil and defense applications classes, and to relate *TCSEC* features and assurances to both civilian and military requirements.

General Threats to Computer Systems

In the defense environment, disclosure is usually judged to be the most significant threat to computer systems. Public law requires the protection of certain kinds of information, especially that related to the national defense. For truly sensitive information, individuals are allowed the discretion to choose to whom they will release information only within the narrowly prescribed limits of security clearances. It is considered serious when information is disclosed to a person with an appropriate clearance but without the necessary need to know for the information. Because such recipients have been properly investigated and are trusted to protect similarly sensitive information, the disclosure is not considered extremely severe. However, if information is released to an unauthorized, untrusted person, the disclosure is deemed very serious because it may be tantamount to disclosing the information to a hostile agent. Through such a disclosure, physical assets may be lost, a competitive advantage may be compromised, or an expensive recovery may be occasioned.

In a medical example, sensitive information may be protected by being divided into several groups: physician, accounting, patient record, statistical, and so forth. Within each group, some individuals will have the right to see more information than others. The physician may have some notes that are strictly for her reference, others to be shared with colleagues in a physician group, others to be shared with the patient, others to be shared with lawyers or with insurers, and others to be shared with the public at large. There is a definite reason for creating each of these subdivisions, due to such factors as patient confidentiality and personal or professional liability. If confidential patient information were released to a public database or a physician's private notes were erroneously made available to a patient, there could be grounds for a successful lawsuit with a high judgment. In a corporate setting, different groupings of data might include wage and salary, other personnel, profit and loss, projected sales, research and development, product strategy, and pricing data. Within each of these groupings different people may have access to different sets of information: some accountants need to be able to see all salaries, while others need to see salaries of only a certain group.

Many large companies with highly-valued information assets have established rigid hierarchies of data: one class is freely released; one class is freely released inside the organization, but only to certain outsiders (e.g., those who agree to protect proprietary data); one class is not released within the organization at large, but is released to members of a select group (e.g., members of the accounting department or of a particular project's staff); and another class is never released to outsiders or to members of other groups in the organization.

There are two basic classes of disclosure: release to an unauthorized person within the organization and release to an outsider. Release to an unauthorized person within the organization may be a more serious problem than in a similar military situation, because not all persons in the organization can necessarily be trusted to preserve confidentiality, and because for some pieces of data concern persons in the organization. For example, the release of salary data to employees could have a serious negative effect on morale, as could the premature disclosure of the closing of a division, or worse, a plan for the closing of a division, where the plan was rejected without having been publicized but was never purged from the system. The "cost" of unauthorized disclosure ranges from a lawsuit (for revealing personal data) to loss of employee or customer confidence, to loss of competitive edge, to loss of ability to function.

There is no commercial counterpart to the military clearance. Individuals' requirements to access data depend on the nature of that data, the person's job requirements, and various unwritten standards for access, such as time with company, experience, level of responsibility, perceived company loyalty, relationship to company (e.g., employee, consultant), and so forth. In an organization, if access is allowed by some individuals and not by others, a *de facto* clearance situation exists. However, there is little formal codification of the standards by which such a clearance is conferred, the expected behavior of the individual holding such a clearance, or the transferability of such a clearance to other items of data. It is partly because of this lack of formal clearances or rigid clearance-granting procedures that civil sector computer system security administrators think the military security model and systems developed for military uses are inappropriate for civil needs.

Untrusted Computer Systems

An untrusted computer system is a system that is either known to possess exploitable security flaws, or that has not been evaluated against accepted evaluation criteria. In untrusted computer systems, it must be assumed that any user having access to the system can read any file in the system, alter or write over any file in the system, and execute any program in the system. Because they cannot reliably prevent any user from accessing the information of any other user, untrusted computer systems may be safely operated only in *a mode dedicated to a well-contained user community*. In general terms, no user should be permitted to access an untrusted system if there is any information anywhere in the system that should not be freely available to that user. Conversely, no information should be entered into an untrusted system if the originator of the information objects to making that information freely available to all users in the system.

In military terms [DoD28], this *dedicated security mode* requires that all users have the clearance or authorization and need to know for all information handled by the computer system. If the system possesses special access information, all users require special access approval. In the dedicated mode, an automated information system may handle either a single classification level and/or category of information or a range of classification levels and/or categories. However, once information is entered into a dedicated mode system, it must be handled and accounted for according to the rules that apply to the most highly classified information in the system. Treatment of any output from the system as less sensitive than the most highly classified data that is processed by the system requires manual review and reclassification of the output.

In civilian terms, if the owner or manager of an untrusted system wishes to protect any information in the system from anyone at all, the system should be physically protected from unauthorized access and should not be connected to publicly accessible communications media. Physical protection means placing the system in a locked room or under appropriate surveillance or other control. For physical protection purposes, the *system* includes all computer hardware and storage media (program and data tapes and disks). It should be stressed that the software in untrusted systems may contain code that displays, prints, or transmits any information in the system via any input/output port at any time. It may be extremely difficult to determine if the software contains such unexpected functionality, or whether such illicit activities are underway at any particular time. The software may also contain code that corrupts any program or data generated by or resident in the system.

Because the only individuals who may be granted access to an untrusted system are those individuals who may be privy to all information on the system, organizations that have policies restricting access to some information (e.g., salary information, tax records, acquisition plans, psychological profiles) are often forced to dedicate a computer system to each category of restricted information. Note that it is not sufficient for users of sensitive data to maintain that data on physically protected and controlled media (e.g., removable disks). If another user of the system wants the information contained on the removable media, he or she has only to add to the untrusted operating system any desired data process that will copy to an accessible medium. Similarly, the system cannot be connected to a network that has some subscribers not authorized to handle all of the information processed by that system. Network access to an untrusted system can be made to result in access by anyone on the network to all information in the system. Thus, the untrusted system that processes sensitive information cannot safely partake of the advantages of connection to a network database participate in network transactions.

Trusted Computer Systems

As defined by the TCSEC, trusted computer systems control, through use of specific security features, access to information such that only properly authorized individuals, or processes operating on behalf of properly authorized individuals, will be permitted to read, write, create, or delete information. The TCSEC identifies six fundamental requirements of trusted systems:

> *Security policy*—There must be an explicit and well-defined security policy enforced by the system. That is, there must be a set of rules governing which user processes are permitted access to any specific computer memory element or device.
>
> *Access control*—Access control permissions must be associated with objects[1]. In the case of systems in which access permissions are specified by a system security administrator rather than by individual users, access control *labels* must be associated with objects. That is, it must be possible to associate every object with a label that readily identifies the access restrictions associated with the object.
>
> *Identification and authentication*—Individual subjects[2] must be identified. That is, users or programs acting on behalf of users must be unambiguously identified to the operating system's protection mechanisms. Identification of users and programs associated with those users is a necessary prerequisite to determining whether or not a requested access can be permitted.
>
> *Accountability*—Audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party.
>
> *Assurance*—The computer system must contain hardware and software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces the foregoing policy and accountability requirements.
>
> *Continuous Protection*—The trusted mechanisms that enforce these basic requirements must be continuously protected against tampering and/or unauthorized changes.

[1] The TCSEC defines "objects" as passive entities that contain or receive information. Examples of objects are: records, blocks, pages, segments, files, directories, directory trees, and programs, as well as bits, bytes, words, fields, processors, video displays, keyboards, clocks, printers, network nodes, etc. Access to an object potentially implies access to the information that it contains.

[2] According to the TCSEC, a *subject* is an active entity (generally in the form of a person, process, or device) that causes information to flow among objects (e.g., data or devices) or changes the state of a system.

Discretionary and User-Specified Access Control

Systems that provide discretionary protection must enforce a policy under which an authorized user establishes, and the system enforces, criteria for the access to any objects under the user's control. Typically such a policy is expressed in terms of access control lists that identify by name or by some defined grouping all subjects that may access a specified object. All authorized subjects must be identified to the system at a granularity consistent with the requirements of the security policy. It is essential to require that the protection system operate in a domain that protects it from external interference or tampering; without such a requirement, the protection qualities of the system cannot be assured. It is also essential that there be some degree of confidence in the correct implementation of the protection mechanism; such implementation assurance can be achieved through testing, through description of the system features, through documentation of the system's design, through inspection, or any combination of these. The minimal feature and assurance requirements of discretionary protection systems are summarized in Table 1.

Table 1: Discretionary Protection Summary

Features		
	Policy	Authorized user controls access to all objects under user's control; objects being controlled, modes of access control, classes of accessing subjects to be specified.
	Access Control	Implicit in access control mechanism; associated with each controlled object.
	Identification & Authentication	Sufficient to implement granularity of security policy.
Assurances		
	Accountability	As needed to document functioning of security enforcement mechanism.
	Assurance	Combination of design, testing, documentation and inspection of implementation, implementation environment requirements.
	Continuous Protection	Separate domain of execution.

Discretionary access control, which uses access control lists (or an equivalent), is known to suffer from a fundamental weakness: protection is afforded to the container, not its contents. Thus, user A can assign access to file F to users B and C only; however, B make a copy of the file (for example, by reading the file and writing a new one which is the duplicate of F), and allow any other users, including ones that A did not intend or would have rejected, to access the copy of F. Thus, the discretionary protection mode depends largely on the discretion and judgment of all users to act in a way consistent with the security objectives of the system.

Under discretionary access control rules, read access, write access, and execute access permissions are normally specified independently of each other. On most systems, a user who has permission to read a file can also make a copy of the file. In the user-specified or DAC security mode, the originator of information should not grant permission to any other user to read that information unless the originator trusts that user to restrict access of any copies made of the information to the set of users that were granted access to the original version of the information. Conversely, no user should be granted access to a DAC environment who cannot be trusted to perpetuate the access restrictions placed on information by its originator.

As shown in Table 1, the features required of a discretionary protection system need only implement the security policy. For a given situation, it may be acceptable within the policy to identify users only by group or class, or to provide protection for only certain objects, or to limit only certain types of access (e.g., no control on reading but limitations on writing). Similarly, the assurances are minimally those to ensure that the system is implemented as specified, and does properly perform its specified function. As examples of the discretionary protection class systems, *TCSEC* C1 and C2 systems meet a reasonable set of requirements for features and assurance, although in other contexts, it may be appropriate to levy more or less stringent requirements on systems to enforce discretionary security. *TCSEC* requirements for C1 and C2 systems specify relatively few assurance requirements, under the assumption that access control truly critical to the national security will be implemented by stronger mechanisms. Thus, under the *TCSEC*, systems for discretionary access control also tend to be of relatively modest assurance, which causes people to conclude erroneously that systems providing only discretionary access control are of necessity low assurance.

The minimum criteria acceptable to the DoD [DoD28] for operation in other than the dedicated mode is called "controlled access protection." Systems meeting TCSEC requirements for the controlled access "make users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation." These systems also include mechanisms that prevent reassignment of a medium (e.g., page, frame, disk) to a new user or process that contains residual data entered by a previous owner. Computer systems that meet controlled access protection requirements can reliably enforce user-specified restrictions. Evaluation criteria for higher level (i.e., C2) systems include requirements for audit mechanisms that record successful and unsuccessful attempts by each user to access information. If a discovery is made that copies of information have been distributed to users not granted access by the originator of a set of information, the audit mechanism provides a means for identifying the user or process responsible. Again, these additional requirements of C2 over C1 reflect the case that operational requirements may affect the requirements for a particular class of systems.

In military terms, the user-specified or discretionary protection mode of operation is known as the "system high" security mode. [DoD28] The system high security mode requires that all users having access to a computer system possess a security clearance or authorization, but not necessarily need to know, for all information handled by the system. As in the dedicated mode, an automated information system may handle a single classification level and/or category of information or a range of classification levels and/or categories. Again, once information is entered into a dedicated mode system, it must be handled and accounted for according to the rules that apply to the most highly classified information in the system, and any output from the system can be treated as less sensitive only after manual review and reclassification of the output. It is assumed that all military users granted access to a military system high environment can be trusted to perpetuate the access restrictions placed on information by its originator.

In most user-specified or system high environments, a user can restrict the community of users who can read, alter or write over, or execute programs he or she originates. However, once the user grants access permission to another user, the user loses control over distribution of copies of that information. If the information is a program, control is lost over which users or processes can execute copies of that program. Any restrictions regarding altering or writing over the original copy of the information (or executing the original copy of a program) remain in force. This may suffice for some cases in which the security objective is not confidentiality but the integrity of files or programs.

Administrator-Based Access Control

In many cases, computer system managers need more rigid control of access to information processed by the system than is available from user-based access control. It may be desirable to require that a system security administrator determine, based on organizational policies and procedures, which users are to be granted access to special categories of information (e.g., payroll information, proprietary research findings, marketing plans, competition-sensitive financial data, psychiatric and other medical records). In order to operate safely in this mode, a system must possess all of the attributes found in controlled access systems, plus provide for assignment of sensitivity tags to all files and devices, association of authorization levels with all user identity/password pairs, be based on a statement of the security policy model of access to objects by subjects, enforce mandatory access control over accesses of subjects to objects in accordance with the stated model, accurately label exported information, and have removed any flaws identified by security testing. The TCSEC describes this class of system as providing *labeled security protection* (B1). It is noted that labeled security protection criteria does not require access mediation for all objects in the system.

Under administrator-based access control, users are granted permission to access classes of information that are identified by named tags or labels. When users are registered with a system, they are authorized access to information having certain specified labels. (When devices are installed, they too receive security labels.) At the time of log in, a user specifies the label of the class of information to be accessed as well as an identification and an authenticating password. The system security mechanisms ensure that the user is authorized access to the category of information specified before the user can proceed. All information entered by the user is labeled by the system security mechanisms with a sensitivity label that matches the label entered by the user at log in. The system also ensures that no data is sent to any output port unless the system security policy permits information having the label associated with the data to be sent to a destination having the security label possessed by the port. This label-based access control is also known as mandatory access control (MAC). Typically, systems that provide no more than labeled security protection (TCSEC B1 systems) are employed in applications lacking a policy-based hierarchy among labels.[3] The requirements of administrator-based access control are summarized in Table 2.

Table 2: Administratively-Based Access Control

Features	
Policy	Policy established by an authority higher than the user. User may also have some discretion—within bounds of higher authority's policy.
Access Control	By labels implicit in access control mechanism; associated with each controlled object.
Identification & Authentication	Needed for each subject, in order to match subject's identity with administratively-defined access rights.
Assurances	
Accountability	As needed to document functioning of security enforcement mechanism, and to demonstrate proper enforcement to higher authority.
Assurance	Combination of design, testing, documentation and inspection of implementation and environment.
Continuous Protection	Separate domain of execution.

The U.S. military has identified a special mode of operation for relatively low assurance (B1) systems that provide mandatory access control. This compartmented or *partitioned* security mode defines operations wherein all personnel have the clearance, but not necessarily formal access approval and need to know, for all information handled by the system. [DoD28] This partitioned mode requires that any information classified below the maximum level processed in the system be handled as and accounted for in accordance with the rules applying to the most highly classified information in the system. Partitioned mode does, however, permit organizations to avoid the requirement to grant all system users authorization for all non-hierarchical categories of information processed by the system.

Multilevel Operation

Where security domains are structured hierarchically, computers that permit a system administrator to establish restrictions on access to classes, or domains, of information by system users and are capable of reliably enforcing those restrictions are said to operate in the "multilevel security" mode. In military systems, there is an established hierarchy of unclassified, sensitive, confidential, secret, and top secret information. Categories, as described above, may exist within some or all of the hierarchical levels. In many civil environments, each *category* of information may have its own internal hierarchy. In either case, the level of trust provided by the labeled security class of the TCSEC may not be adequate to be safely employed to separate information at different levels within a hierarchy. A reasonable minimum level of assurance for multilevel operation is provided by the

[3] Although some category labels may be assigned to include a group of other labels.

structured protection (B2) requirements of the TCSEC. The military considers structured protection to be the minimum level of trust acceptable for separating classified information from unclassified information. [DoD28]

The policy of a multilevel mode system functions under policy requirements very similar to those of administratively-based access control: an administrative authority establishes a policy for marking all data, for assigning subjects to access classes, and for controlling the access of these subjects to these objects. All objects to be controlled must be labeled and all subjects to have access must be assigned to an access class. In order to be sure of proper association of individuals and classes, it is necessary to provide individual identification and unambiguous authentication. Audit is used to demonstrate that the system is functioning properly, both to the users and to the higher administrative authority; audit is also used to identify and assess the damage from access control failures, either from a system, human, or administrative error. The necessary confidence in the correct implementation and continuous functioning will depend on the seriousness of the administratively-based access control policy. The requirements for multilevel secure implementation are listed in Table 3.

Table 3: Multilevel Mode of Security

Features		
	Policy	Established by an authority higher than the user. User may also have some discretion—within bounds of higher authority's policy.
	Access Control	Labeling implicit in access control mechanism; associated with each controlled object.
	Identification & Authentication	Needed for each subject, in order to match subject's identity with administratively-defined access rights.
Assurances		
	Accountability	As needed to document functioning of security enforcement mechanism, and to demonstrate proper enforcement to higher authority.
	Assurance	Combination of design, testing, documentation and inspection of implementation, implementation environment requirements; because of requirement to separate two or more classes of users and data, relatively strong assurance of correct implementation is necessary.
	Continuous Protection	Separate domain of execution for protection mechanism; because of requirement to separate two or more classes of users and data, relatively strong assurance of continuous correct functioning is necessary.

In military terms, the multilevel mode of operation allows two or more classification levels of information to be processed simultaneously within the same system when not all users have a clearance or formal access approval for all data handled by the system. The multilevel secure mode of operation allows information at different classification levels to exist concurrently on an ADP system where not all users are cleared for the highest level of information processed. All multilevel systems should provide mandatory access control, but not all systems that provide MAC may be accredited for the multilevel security mode. The degree of assurance of a system's security enforcement mechanisms, as evaluated against established criteria, determine the range of security levels for which the system may be accredited. High assurance (A1) systems can cover a range of classification levels from Confidential to Top Secret with compartments, while lower assurance (B1) systems can cover only the range of Top Secret with one or more compartments to Top Secret with one or more compartments. Clearly users holding less than Top Secret clearances are not allowed access to Top Secret data. The advantage of a multilevel system is its ability to segregate different classes of users.

In the *TCSEC,* discretionary access control (DAC) is used for the less rigid part of the security policy, and MAC is used for the more rigid part. For this reason, DAC enforcement is sometimes perceived as less important than MAC enforcement, and so greater assurance of correctness is required for MAC implementations than for DAC. In one sense this is a very reasonable position, because significant confidence is required in the ability of a system to support multilevel mode operation, which means to separate lower clearance users from higher sensitivity data. However, in another sense, lower assurance is not necessary for the implementation of DAC. That is, one might want to implement a very high assurance DAC system. As is shown in Table 1, Table 2, and Table 3, in principle, assurance levels can be adjusted as appropriate for a particular system; while it is probably unreasonable to implement a low assurance multilevel system, it is not unreasonable to implement a high assurance user- or administrator-based access control system.

An example of the utility of high-assurance DAC is as follows. One government agency (not the DoD) publishes summary statistics derived from private sources. The statistics become releasable in the aggregate, although the components from which the statistics are derived are highly proprietary. Different agency employees must work with specific figures. It may be possible to implement this control using a careful but artificial scheme of levels and categories using mandatory access control. However, controlling access to this data by individual access control lists is probably both more natural for the users and more compatible with the overall security policy being enforced. For this application, which is not unlike the needs of many civilian organizations, a high assurance form of user-based access control is very desirable.

Notion of Hierarchies and "Risk Range"

For systems managing hierarchies of security labels, the U.S. National Computer Security Center (NCSC) has established a concept, called the "risk range," by which to evaluate the potential exposure of handling data on computing systems. The range relates the sensitivity of the data, the clearances of the personnel who use the system, and the conditions of the system's development and use. Given the level of the most sensitive data on the system, the maximum clearance of the user with the lowest clearance, and the type of the system, a chart exists by which to determine the minimum degree of trust required for the system. [CSC004]

The rationale for the selection of these minimal requirements includes both features and assurances. For military security, implementation of DAC, which is based on the security classification, requires features such as labeling to enforce the separation of users at one clearance level from data at another. Therefore, any situation in which a system will operate at two or more levels concurrently requires labeling of all objects visible to the user, which necessitates a system rated B2[4] or above. To enforce discretionary access control (DAC) requires only the ability to maintain and use lists of subjects allowed to access a single object, which occurs at the C1 level. In other cases, the choice of minimum trust is based on a less precise notion of trust, which equates to assurance of correctness. Thus, a system that supports users some of whom are cleared to only the Confidential level, with Top Secret data having multiple compartments, in a closed environment, is said to required an A1 system. In this situation there are no specific features that would require a system higher than B2, but the degree of risk has been judged more significant than separating Confidential from Top Secret with no or one compartment, and so the class of systems required rises from B2 to B3 to A1 for these three cases (none, one, or more than one compartment). These requirements are minimum requirements, with the caveat that a higher required evaluation class might be required if, for example, there is a high volume of data at the maximum data sensitivity, or there are many users with the minimum clearance.

As the previous paragraph points out, the minimum acceptable evaluation class of a system for a particular situation is a somewhat subjective judgment, based on the kinds of data being processed, the kinds of users, and the kind of system. The minimum evaluation class depends both on certain required features and on the ability to ensure the system's correctness.

Commercial computer security requirements are somewhat different, in that the negative effect of some forms of unauthorized disclosure can be significantly worse. It is important to maintain separation not just by department but also by job function and need-to-know. Therefore, both features and assurance of correctness are important in a selection of trusted systems for the commercial environment.

In the commercial environment, it is necessary to have a means of identifying data that relates to the accounting department versus that which belongs to the personnel or management group. Such a division is naturally implemented by MAC, and so

[4] Strictly speaking, B1 systems provide mandatory separation by classification level. However, these systems are missing two important qualities: first, the mandatory access control need apply only to a defined subset of the objects in the system, so that it is not necessary that every object of the system be under mandatory access control; and second, B1 systems lack the architecture that gives high assurance to the inviolability and completeness of the trusted computing base. Therefore, a B2 system is the minimum system that can provide highly reliable separation of object by level.

a system with at least B2 functionality is necessary. If information is also to be divided within departments, that is, by job function and need-to-know, such separation can be implemented via a more complex class structure (by using a large lattice of incomparable categories) or by using discretionary access control. Each of these means of implementation has advantages and disadvantages.

Relationship between Military and Civilian Multilevel Applications

It is often alleged that only the military community has hierarchical or level-based information protection requirements. This simply is not true.

The military scheme for control of information dissemination is based on classifications of information and clearances of individuals. Because this one scheme is used throughout the military and intelligence community, it is well codified (supported by Federal law), and widely understood. The typical reaction from the civilian sector is that the rigidity of the military scheme is excessive, that civilians do not have civil-sector clearances, and that division into classification levels such as Secret and Top Secret is incompatible with the true civil-sector needs. In fact, the underlying framework of the military information protection scheme corresponds very closely with civilian information protection practices.

Consider, for example, a corporation's data handling needs. The accounting group handles financial information, the personnel department handles confidential employee data, the sales staff handles sales projections and marketing leads, and the research group handles data related to future corporate products. Confidentiality for some employee data (e.g., social security number) must be preserved by law, while other data (e.g., individuals' salaries) is protected as part of a bond of employer-employee confidence. The names and status of corporate customers and prospects are closely held in a situation in which there is significant competition. Finally, product development plans represent the future marketability for the company. These divisions between data are typically considered extremely important. Naturally, there is some sharing, for example between personnel and accounting organizations (so that salary changes can be implemented), but the kinds of data shared and the roles of individuals sharing that data are part of an administrative policy. Largely, the individual groups are treated as separate divisions, very much like compartments in the military model.

Furthermore, within a group, not all data will be accessible to all employees. Certain personnel clerks may be allowed, because of their job functions, to access salaries of all hourly workers, but not of the salaried staff. Some clerks have no access to wage or salary data. Some clerks have the ability to see (read) but not modify (write) the data. Some high-level management have the ability to see all personnel data. Thus, there may be levels within compartments, ranging from No Wage or Salary, to Read Wage Only, to Read Wage and Salary, to Read and Modify Wage and Salary.

Functionally, this situation closely resembles the military model with the most significant distinction being that compartments are considered first, and then there may be a hierarchy within compartments. In spite of this apparent difference, there is essentially no functional difference (from the perspective of protection of information) between compartments on top of levels or levels within compartments. Pictorially, the military classification scheme is typically viewed as a four-level hierarchy. It may be more appropriate to view the civilian information protection scheme as a set of largely disjoint groups, with, possibly, levels within the groups. Under this interpretation, there will be a few high-level corporate managers who can access all information from all groups. There will also be some information that is accessible to all persons in all groups (for example, the kind of information typically found in the company newsletter). The civilian sector representation of sensitivities of data is shown in Figure 1. The compartments of information are represented as ovals, and levels of data within each compartment are shown by lines. It is not necessary that there be any correspondence between the lines of one compartment and those of another.

This organization is not limited to a corporate structure. Medical and other professional groups have similar hierarchies. A patient's record may consist of several portions: billing information, statistical [anonymous] disease control data, a portion shared with the patient, a portion shared with other physicians in a shared practice, a portion shared with other consulting physicians, a portion shared only with the physician's lawyers, and a portion private to the attending physician alone. There seem to be two orthogonal sets of community of interest: first, the accounting group needs access to the billing-related information for all patients, but consulting physicians need access to the consulting-related data for only those patients on whose cases they consult. In fact, the consulting physician data may be better handled as a need-to-know issue. Then, the information protection needs are broken into compartments by function (accounting, consulting physician, and so forth), and any single patient's record is further protected within a compartment by need-to-know, when necessary.

Trusted Systems for Civil Applications

As the examples just described show, there are strong similarities between the information protection needs of the military and civil sector. To date, most development of trusted systems has been in response to military requirements, and the terminology of trusted systems development has tended to reflect this genesis. Thus, today's trusted systems support "classification levels" and "compartments," even though these terms are unfamiliar to, or at least thought to be inappropriate for, the civil sector. However, as described above, the civilian sector today functions with strong divisions between classes of data and with a sort of hierarchy of access within each division. It is very straightforward to map civilian needs onto military trusted systems. This situation is most fortunate for the civil sector, because it means the military environment has paid for the research and development necessary to cause the production of systems with both the features and the assurances needed by the civil sector. With little or no work, these same systems can be made available to meet civil sector needs.

The civil sector does not currently have accepted criteria for the evaluation of trusted systems, nor does it have accepted guidelines on the use of such systems. However, here, too, the military criteria and guidelines can be tailored for civil sector use.

As described previously, the features of the *TCSEC* are fundamentally the same features that are needed for separation of data into distinct divisions in commercial systems. The primary feature needed is mandatory access control. For the military model, the standard way to organize data is to first divide it into various hierarchical levels. Non-hierarchical categories are then (conceptually) overlaid on each level, thereby producing data that is Secret-Compartment A or Top Secret-Compartments B and C. In the civilian model presented in this paper, it is more appropriate to conceptualize data as having been divided into distinct groups based on function (e.g., personnel, sales, administration, accounting, consulting-physicians, statistics, and so forth). These groups are non-hierarchical, just like the compartments of the military model. Within each group, access is allowed to senior administrators, managers, junior administrators, and so forth on a hierarchical basis. Thus, whereas the military model seems to resemble hierarchical levels overlaid with non-hierarchical categories, the civilian model more closely resembles non-hierarchical groups overlaid with a hierarchical level-of-authority structure. In fact, it makes little difference whether access by hierarchical component is determined before or after the non-hierarchical component, as long as both are checked before success or failure is reported to the user.

Summary

This paper outlines the modes of use of untrusted and trusted computer systems, the functionality and assurance requirements for design of such systems, and the possible uses of such systems. The needs of the civil sector are frequently thought to be very different from those for the military community; however, this paper shows that both have similar—and more importantly, compatible—requirements for the protection of information, and that these requirements lead to similar implementation approaches. In particular, the qualities of multilevel systems (those rated B2 and higher under the *TCSEC*) are applicable in civilian settings as well as in military ones, with appropriate understanding of security policies for civilian applications.

References

[CSC003] U.S. National Computer Security Center, *Computer Security Requirements — Guidance For Applying the* Department of Defense Trusted Computer System Evaluation Criteria *in Specific Environments*, Document CSC-STD-003-85, 25 June 1985.

[CSC004] U.S. National Computer Security Center, *Technical Rationale Behind CSC-STD-003-85:* Computer Security Requirements — Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, Document CSC-STD-004-85, 25 June 1985.

[DoD28] U.S. Department of Defense, *Security Requirements for Automated Information Systems*, Department of Defense Directive Number 5200.28, 21 March 1988.

[TCSEC] U.S. Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria*, Department of Defense Standard 5200.28-STD, December 1985.

Executive Summary

Implementation of the Computer Security Act of 1987

Dennis Gilbert

National Institute of Standards and Technology

By establishing Public Law 100-235, the Computer Security Act of 1987 (the Act), Congress enacted a measure for establishing minimum acceptable security practices for federal computer systems that contain sensitive unclassified information. The Act places major emphasis on computer security planning. A significant aspect of the first'year activities under the Act included the review of federal agencies' security plans which were submitted to a joint NIST/National Security Agency (NSA) review team. Current instructions from OMB shift emphasis to the implementation of computer security plans. This strategy provides for visits by OMB, NIST, and NSA staff. The group will provide direct comments, advice, and technical assistance relative to the agency's implementation of the Act.

Private sector organizations can also learn and benefit from the federal experience in implementing the Act. Although federal managers have specific regulatory requirements that must be satisfied, much of their data processing and security needs and perspectives are similar, or directly analogous, to their counterparts in the private sector.

This session will present different perspectives on the implementation of the Computer Security Act from representatives of organizations that have played key roles in the process. Emphasis will be on the learnings that have come out of the efforts to dates. Also discussed will be current and future directions under the Act.

THE CSO'S ROLE IN COMPUTER SECURITY

Cindy E. Hash
National Computer Security Center
Ft. George G. Meade, MD 20755-6000
(301) 859-4360

ABSTRACT

This paper addresses the role of the Computer Security Officer in computer security from two perspectives. How a Computer Security Officer can use the Department of Defense Trusted Computer System Evaluation Criteria, better known as "The Orange Book," to gain an understanding of his/her responsibilities is one perspective. By using the Orange Book as a guide, a CSO can create a good set of responsibilities. The other perspective deals with the relationship that exists between a Computer Security Officer and a computer user. If the two work well together, a would be hacker has a minimal chance of breaking into the associated computer. If either chooses to ignore established security procedures, their computer becomes extremely vulnerable to a penetration.

INTRODUCTION

In this day and age computers have become a major part of everyone's life. Some people are not aware of the enormous impact a computer has on their life. That is, until the computer does not do what it is intended to do. This "malfunction" can occur because of an operational error or because of the maliciousness of an individual (i.e. computer hacker).

Because of the hacker threat, defining, creating and practicing computer security has become increasingly important over the years. In those instances where a computer is shared by more than one user, or when the computer is attached to a mainframe or a network, someone should be responsible for ensuring the security of that environment and guarding against hacker attacks. The often asked question is, who should that person be? Many government and industry groups have delegated that responsibility to the infamous Computer Security Officer (**CSO**). The CSO has also been called the Information Systems Security Officer (**ISSO**) or the Computer Equipment System Security Officer (**CESSO**) This paper will use the term CSO.

Some computer systems do not have a designated CSO. In those instances, enforcement of computer security is often an additional job undertaken by the System Administrator. The security of that system is in good hands if that person is motivated. Dedication is another quality a CSO should possess. Sometimes security problems can take days or even months to resolve.

The CSO is also responsible for enlisting the help of the user population. Users need to be informed of the role they play in maintaining computer security. The efforts of a CSO will be in vain, however, if the user group contains one or more careless users.

Therefore, the security of any system is only as good as the conscientiousness of its' CSO. A good CSO constantly works to maintain and enhance the security of a computer system. In doing so, the CSO accomplishes his/her responsibility to provide the user with a secure environment.

CSO RESPONSIBILITIES

All CSO's are responsible for ensuring system security. Different systems have different levels of security. This brings us to the question, "what is a secure system?". According to the Department of Defense Trusted Computer System Evaluation Criteria, there are **six basic requirements** associated with a secure system.

1. SECURITY POLICY - There must be an explicit and well-defined security policy enforced by the system.
2. MARKING - Access control labels must be associated with objects.
3. IDENTIFICATION - Individual subjects must be identified.
4. ACCOUNTABILITY - Audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party.
5. ASSURANCE - The computer system must contain hardware/software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces requirements 1 through 4 above.
6. CONTINUOUS PROTECTION - The trusted computing base (TCB) that enforce these basic requirements must be continuously protected against tampering and/or unauthorized changes.

Not all systems meet each of these requirements. The CSO should find out which of these requirements apply to his/her system, and then enforce them. What follows is how each of these requirements apply to the CSO.

1. Operate with Sound Security Policy

Without a Security Policy, there are no rules for the users to follow, nor are there any procedures to follow in the case of an emergency. If there are no rules on a system, each user has to rely on the integrity of other users, in order to feel that his/her files are secure. However, not everyone can be trusted. A good Security Policy should include the following information, as a minimum:

a. Definition of the system.
b. The type of information stored on the system.
c. The type of activity allowed on the system.
d. What type of software, if any is to be used in conjunction with the system.
e. The allowable forms of connectivity to the system.
f. A reporting mechanism to follow in the event of a security incident.
g. List of responsible individuals and what their role is in association with the system.
h. A Contingency Plan to be used in the event of an emergency (i.e. fire, flood, etc.).

With this information the CSO knows what needs to be protected, the forms of protection needed and what to do when this protection has been violated.

2. Verify Access Control Labels

The system software should label all objects properly. It is the CSO's responsibility to periodically check to see that objects are being marked appropriately. If they are not, it is possible that someone has tampered with the system software. This should be researched immediately.

3. Ensure Unique User Identification

The CSO should instruct the accounts administrator to use unique identifications for each user. Group user id's should be discouraged. Each user can be held accountable for his/her actions with a unique id.

4. Use Audit Trail Information

Each morning, a CSO should read highlighted portions of the audit trails from previous day's activities. These audit trails should provide the following important information:

a. ANY system code changes.
b. Denied file access attempts.
c. Failed login attempts of *n* or more times.

If any system code has been changed without the consent of the System Administrator and the CSO, the system has been penetrated. At that time the CSO needs to begin an immediate investigation into how the system was violated. The code should be corrected and precautionary measures installed against any further attacks. The audit trail should have indicated what user performed these changes. That user will have to be questioned. If it is discovered that the account was captured by an unauthorized user, precautionary measures will have to be taken for the owner of the account (i.e. change the password or issue a new account).

If there were denied access attempts in the audit trails, certain actions will have to be taken. This information merely suggests that a user is trying to access a file to which he does not have legitimate access. It is possible that the user will have to be closely monitored or the user may be questioned.

If there are failed login attempts on an account, and if they exceed the limit set by the CSO, the user will be called. If the user did not generate these login attempts, the connection may have to be traced back to its' origin. In any case, this commitment gives the user a sense of security; knowing that someone is guarding against account attacks.

This is perhaps the most important responsibility of the CSO. If the CSO does not view this job as important, the would be hacker has an excellent chance of penetrating a system. However, by daily review of this material, the CSO obtains some pertinent data.

5. Hardware/Software Assurance

While the CSO can not actually perform this requirement, he/she must closely monitor audit trails to ensure that the software protection mechanisms are properly protecting the system. The CSO should also have some sort of policy which prevents unauthorized people from accessing the system hardware. This in turn gives some assurance that no one can tamper with the hardware protection mechanisms.

6. Provide Continuous Protection

This is another requirement that the CSO does not actually perform. However, he/she must closely monitor the audit trails with regard to ANY access or attempted access to the Trusted Computing Base.

ADDITIONAL CSO RESPONSIBILITIES

In addition to the above responsibilities, the CSO should remain available to the users and supply guidance whenever necessary. The CSO should also be well known by the users. If a user does not know who the system security officer is, then the CSO has not done his job correctly. All users should know who to call immediately when a security incident occurs.

The CSO should also be readily available when needed. If a user comes to the CSO with a problem, the CSO should be ready to help. A CSO may not have every law or policy about computer security memorized, but should know where to find the guidance on a variety of subjects. This guidance should be shared with users upon request.

USER RESPONSIBILITIES

The CSO should make the users aware of their responsibilities, while actively carrying out his/her own duties. If a CSO is going to expect the users to play an active role in Computer Security, then it is essential that the CSO practices what he/she preaches!

While the CSO is busy ensuring the secure environment, the user should be practicing computer security. A user should have a daily routine which reflects a dedication to this cause. This daily routine can eliminate a lot of security problems. The following should be practiced each day:

1. Logging in to an account should be done with no one "looking over your shoulder."
2. A terminal with an active session should never be left unattended.
3. If the system seems to behave in an abnormal manner, the CSO should be notified.

Although it is human nature to trust others, it is possible that a co-worker would try and access information that does not belong to them. This could be done as a gag, out of curiosity or to intentionally harm someone. That is why it is best to log into an account in somewhat of a private environment. Some people might watch your login pattern, looking to see if they can guess a password by watching your typing pattern.

With that same reasoning in mind, it is crucial that a terminal NEVER be left unattended with an active session running. This provides a great temptation to that dishonest co-worker.

It is better to be safe than sorry, is sound advice for a computer user in many ways, especially if a user notices something odd. It is better to report an anomaly and hear the possibly insignificant reason behind it, than to ignore it and find out months down the road that the strange occurrence did some irreversible damage to important files.

These few examples are just daily routines! Other responsibilities that the user should actively practice include:

1. Frequent password changes.
2. Protection of passwords.
3. Adherence to the system Security Policy.

It is no surprise to those in the computer security business that the threat of hackers is real. Therefore, the more frequently a user changes his password, the better the odds are against his account being compromised. A good system will force its' users to change their passwords at set intervals. However, all systems do not support that type of application. Therefore, it is the users responsibility to change his/her password at the specified interval recommended by the CSO.

The protection of a password is largely dependent upon the owner of that password. It is extremely important that passwords not be left taped to the terminal or left out in the open anywhere near the terminal. Good sense should also be used when selecting a password. Using words or phrases is never a good idea, especially when it can be associated with the user. It is best to select a password with a length longer than six characters, and perhaps one with a mixture of alpha and numeric characters.

It is also essential that the users adhere to the system Security Policy. If rules are broken, a link out of the chain of security leaves an opening for the would be hacker to enter.

CONCLUSION

No matter how secure an operating system is, without a diligent CSO and a concerned user population, the system remains extremely vulnerable. If audit trails are not read, or if users are careless with their password, a hacker has had half the battle fought for him. It then makes the system easier to penetrate.

However, if the CSO is on the alert and has armed his users with the proper knowledge, and if the users take heed to the advice, the hacker will not make it far into the system, if they get in at all. But, it **does** require a team effort. Both parties are extremely important and should be mindful of their duty. With this knowledge, everyone can contribute to furthering the world of successful computer security.

REFERENCES

1. DoD 5200.28-STD, DoD Trusted Computer System Evaluation Criteria, Dec. 1985.

Implementation and Usage of Mandatory Access Controls in an Operational Environment

Leslie M. Gotch
Honeywell Federal Systems, Inc.
Gotch@.Dockmaster.ncsc.mil

Shawn M. Rovansek
National Computer Security Center
Rovansek@Dockmaster.ncsc.mil

Abstract

The National Computer Security Center (NCSC) uses DOCKMASTER, a Honeywell DPS-8/70 mainframe running the B2-evaluated Multics operating system. DOCKMASTER provides a central electronic facility for technical interchange between NCSC personnel, computer vendors, and the US computer security community on unclassified topics related to computer security. To support this role, DOCKMASTER is used to store a considerable amount of vendor proprietary data. Up until January 1989, this information was protected using only a discretionary security policy enforced by the Multics Access Control List (ACL) mechanisms.

In January 1989, the NCSC began utilizing the Multics Access Isolation Mechanism (AIM) to provide Mandatory Access Controls (MAC) to protect vendor-proprietary information stored on DOCKMASTER. Modifications to standard AIM were necessary to increase the number of compartments in order to adequately separate vendor data (i.e., each vendor has a single compartment). This paper discusses the modifications made to Multics to increase the number of compartments used in the enforcement of its Mandatory Access Control policy. These modifications included revisions to the Trusted Computing Base (TCB). This paper will describe the reason for the changes, the extent of work required to make the changes, the adjustments made by users to utilize AIM, and the impact of the changes on user productivity.

Introduction

DOCKMASTER must be readily accessible to authorized users in order to fulfill its mission and is therefore accessible via dial-in lines, TYMNET, and the Internet. Because of its wide connectivity and the proprietary nature of much of the data located on it, DOCKMASTER must be able to prevent unauthorized access to data. Multics provides support for both discretionary and mandatory access controls which can prevent unauthorized access to data.

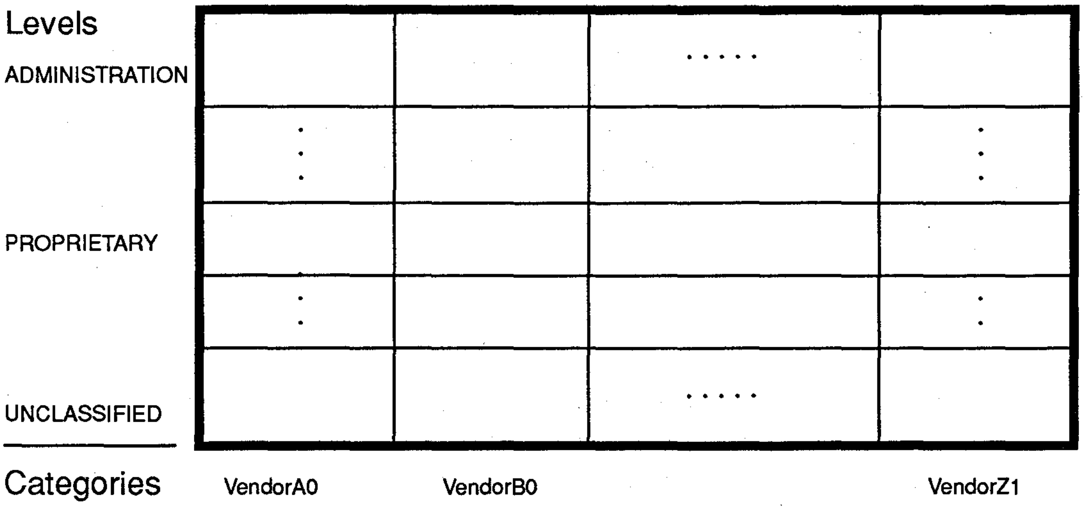
Multics provides Discretionary Access Control (DAC) via Access Control Lists (ACLs) associated with each object. The ACL defines what type of access the listed users and groups of users have to the object. For directories, Multics defines access in terms of status ("s", the ability to view attributes of objects within the directory), modify ("m", the ability to modify attributes of objects within the directory or remove them entirely), and append ("a" the ability to introduce objects into the directory). For segments, Multics defines read ("r", the ability to view a segment), execute ("e", the ability to transfer control to the segment), and write ("w", the ability to add data to the segment) permissions. Additionally, a user can be given "null" access to both segments and directories which precludes that user from accessing the object.

ACL examples

directory		segment	
sm	Bullwinkle.*.*	rew	Rocky.SysMaint.*
sma	Rocky.SysMaint.*	null	Bullwinkle.*.*
sa	*.Multics.*	rw	*.Multics.*
s	*.*.*	r	*.*.*

Each ACL term consists of the type of access allowed followed by a tuple defining the user, project (group), and process instance (e.g., interactive, absentee). The "*" delimiters in the ACL entries denote wildcard characters. Thus, the ACL term "rw *.Multics.*" grants both read and write access to the segment for any user logged into an account registered on the Multics project.

Mandatory access controls on Multics are enforced by the Access Isolation Mechanism (AIM) and can be represented by a grid of 8 hierarchical components called levels and 18 non-hierarchical compartments called categories. The levels are used to segregate data from lowest to highest in order of importance. On DOCKMASTER, the levels range from UNCLASSIFED (lowest) to PROPRIETARY (middle) to ADMINISTRATION (highest). The categories allow the user to separate data into compartments within the hierarchical levels.



The protection provided by AIM could not be used by the NCSC's Trusted Product Evaluations division to separate vendor data because of the insufficent number of categories supported. The evaluations division wanted to assign each vendor a unique category in order to separate proprietary data using AIM. Since evaluations were in process with more than 18 vendors at the time and it was planned to increase this total, the standard 18 categories were not enough. In 1987, the evaluations division approached the DOCKMASTER systems programming staff and asked if there was a way to modify the existing software to accommodate more than 18 categories.

An initial review of the programs involved showed that the 18 categories were defined as 18 bits that could be turned on and off (changed to a one or a zero) independent of one another to form combinations of independent categories. Further review showed that there were 18 additional unused bits that could be used in a similar manner. However, the total of 36 bits used to define 36 categories would still have been insufficient. The evaluations division needed 250 categories to fulfill their requirements.

Finding a Solution

An implementation of the required 250 categories in a manner consistent with the way Multics MAC worked (i.e., independent categories) would have required a major rewrite of the Access Isolation Mechanism. This would have had serious implications regarding the B2 evaluation of the system. An alternative was discussed that would leave the existing system alone and use the 18 "spare" bits as a means for establishing a different type of compartment. These new compartments would be represented by a pattern of bits instead of just one bit. By using patterns of 9 bits, the number of combinations available would be greater than 48,000. These patterns were created to define what became known as "extended categories" to distinguish them from the regular categories of one bit each. This modification left the AIM authorization checking procedures largely intact.

It became readily apparent that there were going to be some shortcomings with the extended categories. Each category would have to be mutually exclusive, meaning that a user could only be assigned to use one extended category at a time. The use of two extended categories could possibly create a bit pattern that would be a superset of the two being used. This would allow inadvertent access to categories not intended for the user. In fact, the potential existed that if two particular extended categories of 9 bits each were turned on at the same time, the superset of these two extended categories would have all 18 bits turned on providing the user potential access to all vendor data on the system. As a result, users who were assigned to more than one evaluation team would have to logout and login at different authorizations each time they wanted to access data from a different vendor.

Implementation

To fully implement these new categories, a method had to be found to name, maintain, and interpret the new categories. One of the prime considerations was to make this as easy as possible, maintain the system integrity, and not introduce any performance degradation. A Multics subsystem was created to act as a table manager in handling these concerns. The total programming effort for this subsystem was one man-month.

Following the creation of the table manager, an extensive search of the system software programs was begun to determine which system programs would be affected by the introduction of the new categories. A total of seven executable programs and two data files needed modification (see Appendix). Three man months were required to make the modifications and to test the changes. These programs were responsible for:

- converting the AIM levels and categories from plain text to machine language and back again
- displaying the AIM level and category information in the system's audit logs
- updating system data bases to recognize the new categories
- allowing the system start up procedures to recognize the new categories
- handling new authorizations at login time

Storage Media Modifications

Prior to running the Access Isolation Mechanism software for mandatory access controls, the system ran in essentially two modes. These were system_low and system_high. System high was limited to jobs such as audit trails and system saves. The users were assigned to do all of their work at system_low.

A new system boot tape was created with the new AIM information on it. Since the system had been running with one level of "system_high", it would not allow the administrators to reboot the system with the new "system_high" classification containing the extended categories. When a reboot of the system was tried with the new system_high on it, it didn't match the system_high on the disk packs from the previous sessions. The system administration staff had to bring the system up to an intermediate level before the comparison was made of the system_highs. A "patch" to the disk headers with the new system_high was made before continuing the boot process. The system staff performed the trial run on a test system several times to make sure of the procedures before attempting it on the production system.

Trusted Computing Base Modifications

Six of the executable programs and both of the data files that were modified are a part of the Multics evaluated Trusted Computing Base. The largest change was to the program that converts the authorizations from plain text to machine language and back again. The major portion of the change was identifying the extended categories, making sure that only one was used, and then validating its authenticity. Several MAC check algorithms for utilizing extended categories were tested to find the most efficient and implemented. The rest of the program modifications were very simple and consisted of one or two line changes. (See the appendix for a summary of actual code changes.)

All of the programming changes were reviewed by the Trusted Products Evaluation staff. They were given original and modified versions of the programs, accounts on the test system where the development was done, and were requested to run tests to verify the actions of the revised software. After their review and approval, the change was implemented on DOCKMASTER.

User Transitions to a Multilevel Environment

Once the decision had been made to utilize AIM on DOCKMASTER and the modifications described previously were completed, the evaluations division management had to decide how to best take advantage of this additional protection. They determined that the best way to use the AIM facilities provided by MULTICS would be to store all evaluation data at its own hierarchical level, PROPRIETARY, above the one accessible to the general user populace, UNCLASSIFIED. Additional isolation would then be provided using the extended categories to separate vendors from one another (i.e., Vendor A's data would be classified at PROPRIETARY,VendorA0 while B's would be at PROPRIETARY,VendorB0).

The Product Evaluations Division then polled the vendors working with the NCSC on product evaluations to determine their preferences for category separation. Several vendors were working with NCSC on multiple product evaluations and many of their personnel were involved in more than one of these efforts. Compartmentalization of each evaluation would cause administrative difficulties in addition to usability concerns. Thus, each vendor was given the option of a single AIM category for all evaluations or an AIM category for each one. All of the vendors initially involved felt that a single category for all of their personnel was sufficient. The System Administrators then created the 32 categories required to support all of the product evaluations in progress at the time.

Next, in order to take advantage of the new AIM categories, the information currently residing on DOCKMASTER at the unclassified level had to be reclassified. This involved upgrading both evaluation reports in progress and evaluation forums and was done on an incremental basis over the course of a month (January 1989). Since each evaluation report had a unique DOCKMASTER directory, it was not difficult for the system administrators to reclassify the whole subtree using privileged commands. Upgrading evaluation forums was more involved. When using AIM, upgraded forums (those above unclassified) had to reside in a directory of equivalent classification. Under the previous method using ACLs, all evaluation forums were within a single directory of meetings labeled as UNCLASSIFIED. Upgraded directories at each extended category were created within the meetings directory. The system administrators then moved the forums to this new directory and reclassified them to the higher level. In order to avoid disrupting normal usage of these forums, links were left behind in the meetings directory so that users would not have to redefine entries in their personal meetings directories. (See the next section for more on personal meetings directories.)

Finally, users had to be provided with some training on how to use the additional AIM facilities. This took the form of presentations to the evaluation community at one of its workshops and of direct mailing of AIM documentation to users. The evaluators could then, based on their own experiences with learning how to use AIM, help the vendor personnel with whom they were working to become familiar with AIM. Within a few weeks, users adjusted sufficiently to using AIM that productivity using DOCKMASTER eturned lar gely to normal.

Effects of AIM on Standard User Data Structures

The standard organization of the MULTICS user environment places many dependencies beyond the user's control on structures stored in the user's home directory. Examples of these user data structures include the value segment, the memo segment, and the meetings directory. The way in which the system attempts to update these structures can cause problems for users who operate at more than one AIM classification on a regular basis.

Each user has a value segment in which things such as default terminal type and printer usage definitions are stored. When a user utilizes a printer for the first time, the system attempts to update that user's value segment. If a user process is at an AIM classification different from that of the user's home directory, the update cannot take place and the print job will not be executed. This problem cannot be avoided. The only way to work around it is to issue commands from the same classification as the user's home directory or to temporarily redefine one's value segment to a segment within the directory at the same AIM classification (typically the user's process directory).

Memo segments contain reminders which a user can set for later display. Typically at log on, a user's start_up program will check the memo segment and print any outstanding reminders. Once printed, the memo segment is updated which causes the same type of problem as the updating of value segments.

The workaround developed for the memo segment problem involved comparing the user's log on AIM authorization with the AIM classification of his home directory. If they differed, the memo segment was copied into the user's process directory (which is always at the user's AIM authorization) and location of the segment redefined to the system. While this alleviated the problem somewhat, it was no longer possible to set memos which would last beyond the duration of the user's current session from any authorization other than UNCLASSIFIED since the process directory was cleared. This problem could not be avoided and was not serious enough to impair a user's ability to perform their job.

Each user has a personal meetings directory residing beneath that user's home directory at the same AIM classification as the home directory. The meetings directory contains links to the forums which the user normally attends. This allows use of a single user-specific search path for all forums. However, since the directory is at the same AIM classification as the home directory, it cannot be updated with new links from AIM authorizations where the forum is accessible. The significance of this problem will be discussed later in the next section.

Impact of AIM on Forum Communication

While the activation of AIM on DOCKMASTER produced many differences in the way users accessed data, the most significant impact occurred in usage of the MULTICS user communications mechanisms such as forum.

In the standard MULTICS implementation, forums are accessible from the single level and category at which they have been created. In order to access a forum under these circumstances, a user must be logged on at the exact level and category of the meeting. While this implementation may be acceptable for most uses, it created significant problems for the product evaluation community. As part of a normal working day, evaluators must monitor activity on forums related to each evaluation in which they participate. Since most evaluators can be assigned to four or more evaluations at once, each using data within a different AIM classification, this would involve multiple logons to simply see if any new transactions were posted and, if so, to read them. This problem was even worse for evaluations division managers who are expected to monitor forums for evaluations for which they are responsible or in which their personnel participate. Thus, a manager might be expected to view forums within every AIM category as part of their normal job responsibilities.

The solution to this problem contained two elements. First, the forum subsystem was modified to allow read access from levels dominating the sensitivity label of any forum (if permitted by the ACL, of course). Next, an extended category called extend_high was created which was a superset of all other categories. By using the extend_high category at the proprietary level in conjunction with forum's new capability of being readable from dominant AIM classification levels, an evaluator or an evaluation manager could check all meetings with a single interactive session. The "supercategory" also provided a way for the product evaluation community to provide separation from all vendor-specific data for its internal discussions of issues pertaining to all product evaluations.

While this solution greatly improved the usefulness of MULTICS running AIM, there were some unavoidable complications. These had to do primarily with maintaining accurate forum usage information and initially attending upgraded forums.

The information maintained for each forum includes a list of participants, the time/date they last accessed that forum, and the transactions they have viewed (denoted by a "seen" switch). When using the "AIM smart" version of forum and attempting to access the forum from a dominant classification level, none of these values can be updated since this would violate the Bell - La Padula *-Property. In the case of the transaction "seen" switches, the inability to update them from a higher level could quickly render the information provided by them useless. In response to queries about new transactions, the system would keep including all of the ones read since the last login at the AIM authorization of the forum regardless of those reviewed at higher AIM authorizations thus preventing the flow of information across AIM boundaries.

Additionally, the first time that a user accesses a meeting, that person's userid must be added to the list of participants. If the user attempts to do this from an AIM classification level higher than that of the forum, a system error occurs. The AIM mechanism cannot permit this because this is also a violation of the *-Property. A user's authorization must exactly match the classification of a forum for the user to access the forum for the first time.

Thus, while some of the obstacles which were side effects of activating AIM could be minimized, users still needed to move between levels on a fairly regular basis. In order to facilitate this, "new_-proc -auth" was activated. (This change occurred in concert with the introduction of Watchwords.) This allowed users to change authorizations by reinitializing their processes without the delay of logging out and logging in again. Thus, users could change levels with minimal delays of under half a minute. This enabled users with a need to work across a wide range of extended categories to view forums or change using the supercategory and perhaps read new transactions from the higher sensitivity level or to "new_proc" to the necessary levels and update "seen" switches while reading the transactions.

By combining the use of an "AIM smart" forum subsystem, an AIM supercategory, and the "new_-proc -auth" option, users working with data under many AIM classifications could get to it with as minimum an effort as possible given the constraints imposed on DOCKMASTER by the Mandatory Access Controls in operation on the system.

While communicating via forum did not suffer significant usability degradation from the utilization of AIM, the addition of links to new meetings at AIM classifications higher than unclassified became a great deal more difficult. The automated facility for adding additional meetings to a user's search paths, invoked via the "fam" command, is no longer usable for meetings at a different AIM classification levels from the invoking user's personal meetings directory. Thus, the only way to update the meetings directory to include new meetings at a variety of AIM classifications is to manually create the link to the forum. While this change did not cause significant loss of usability, it has caused some confusion among users who were used to simply using the "fam" command.

Adjusting to Multi-level Mail

Mail messages are also subject to the controls imposed by AIM. Each message is a distinct object with its own classification label. The classification of a mail message is equivalent to the authorization of the sender. A user's mailbox may only contain messages with classifications between that of the receiver's home directory (UNCLASSIFIED on DOCKMASTER) and the mailbox (which will be PROPRIETARY,category_name for evaluators and vendors in evaluation) inclusive. A user isable to see only those messages in the mailbox which have classifications dominated by the user's current authorization. Thus, a user logged in at PROPRIETARY,extend_high will be able to view mail messages at PROPRIETARY,VendorA0, the user will not be able to reply at the same classification. Using mail with AIM requires users to pay attention to their current authorizations when sending mail to others.

Quota Management

MULTICS provides the facility for controlling how much storage space (in disk pages) a particular directory or hierarchy can use. This is done via allocation of a set quota of disk pages. Quota may be allocated by anyone with "modify" access to both the source and destination directories. Quota placed at the root directory of a subtree will encompass all subdirectories of that subtree at the same AIM classification.

Before AIM was utilized on DOCKMASTER, quota management was not a concern for the average user or for administrators of the various DOCKMASTER projects. With AIM separation in use, users are required to manipulate quota in order to be able to work with proprietary data within their own hierarchies. This created an administrative headache of ensuring that over 400 users had adequate quota to continue normal operations.

In order to solve this problem, the evaluations division project administrators developed programs that automatically executed at 6 hour intervals, checked quota levels on user and unclassified evaluation directories, and added additional quota to them from the root of the subtree as necessary. To manage quota in upgraded directories under the evaluations division hierarchies, the evaluations division project administrators created a privileged program which runs periodically and examined quota levels, reporting any directories nearing capacity. Given these additional tools, quota management at the project level could then be handled with minimal difficulties.

Quota management at the user level required additional training of the type described previously. Once users became accustomed to keeping an eye on quota or to recognizing the error messages resulting from failure to do so, they were able to use DOCKMASTER usually with only some slight overhead of issuing a few additional commands. If the user did not have additional quota readily available, they could quickly exceed the amount of quota allocated to them and might have to wait as long as six hours for the quota watcher absentee to give them more. The only alternative to that would be intervention by a project administrator. Requesting this human intervention thus caused some additional delays in fulfilling quota requests; however, incidents of this type have not proved to be commonplace (averaging fewer than 1 a day).

Impact of a Multi-level Environment

To implement extended categories on DOCKMASTER while maintaining the integrity of the TCB proved to be a challenging, yet manageable, task. While it was nearly two years from first proposal to final implementation, the actual amount of work involved required less than a man-year including coding and testing. Disseminating AIM documentation and providing formal training to evaluators required further efforts on the part of the DOCKMASTER administrative staff which could be measured in terms of man-weeks. Reclassification was the final significant step in transition to a multi-level environment undertaken by the administrative staff. The start-up costs described here proved to be the most significant ones involved in the process.

The effects of AIM transition required some adjustment on the part of the multi-level user community. Disk storage allocated to the evaluations division increased by 35% soon after the transition due to the need to maintain additional storage quota in directories for later usage. Access to each evaluation forum and report directory was not possible for a day while it was upgraded. This impact is most comparable to a temporary outage of network service. Adjustment to the additional commands required to operate at multiple authorizations occurred slowly over the course of the following months without a significant drop in productivity. However, a 3-4% increase in interactive usage among the affected users was noted after the multi-level environment was adopted. This is attributed to the inconvenience of the additional commands required to navigate across multiple authorizations.

The resulting inconveniences created are, however, largely unavoidable due to the need to provide users with an environment in which existence at a variety of AIM classifications is possible. Some of the system constructs which do not support this ability include the values segment, memo segment, and meetings directory. It did prove possible to create workarounds or adjust procedures to make up for this problem. Using forum from dominant AIM classification also caused difficulties. However, altering forum's implementation to allow read-only access from dominant levels, creating a supercategory which encompassed all extended categories, and enabling users to change levels using the "new_proc -auth" command effectively reduced these problems to an acceptable level. The necessity for users to manage their own storage quota and ensuring that additional quota is readily available are probably the biggest adjustments to make in order to use Mandatory Access Controls. These obstacles have been largely overcome through training users how to manipulate quota and the use of automatic quota watchers has, for the most part, ensured that users have additional quota available as needed. After a year of AIM usage, these problems have generally proved to be little more than inconveniences without a significant impact on user productivity on DOCKMASTER.

Conclusion

Creating a Mandatory Access Control environment on DOCKMASTER required significant but manageable efforts on the part of the administrative staff. Using the Mandatory Access Control environment created by AIM utilization has not proved to be a significant burden on users despite some unavoidable inconveniences. As could be expected, using AIM was not popular with users because of these inconveniences; however, feedback from the vendor community has been generally positive toward utilizing AIM.

Appendix: Programs Modified During AIM Revisions

convert_access_class_.pl1

This program converts the ascii representation of an AIM authorization to a bit representation and the reverse process. Additional code was added to perform these conversions for the new extended categories. Also, a new category "extend_high" was added that would allow a user with this privilege to access any of the extended categories at any time.

aim_util_.alm

This program is responsible for checking AIM classifications Entry points are defined within this program to perform different variations of the access checks that may be required. Arguments consist of the bit representation of the AIM classifications to be checked. The program is written in assembly code. Changes were made to modify a mask that checks the bit patterns of the classifications. The mask size was increased to check the extra 18 bits for the extended categories.

display_access_class.pl1

This program displays the AIM classification in a numeric form L:CCCCCC where L is the AIM Level and C is an octal representation of the AIM category bits. This program is called by the logging routines to minimize the amount of information placed in the logs. This program was changed to make the representation take the form of L:CCCCCCCCCCCC so that the extended category bits could be shown.

system_info.pl1

This is the utility program that gathers information about the system. Entries in this program allow other programs to obtain information including the list of AIM levels and categories. Changes had to be made to recognize the extended categories.

ed_installation_parms.pl1

This program is used to modify all system parameters. One of these parameters is the maximum AIM classification that can be used by the system. Changes allowed the maximum authorization to include the extended categories.

as_init.pl1

This program is responsible for initializing the system environment during system start_up and locating and validating various system tables. Changes had to be made to tell it where to find the table that contains the extended categories.

sys_info.cds and pds.cds

These data segments are compiled are used to store various system parameters and act as templates for other functions. These parameters are determined when the operating system is built. Changes had to be made to make the maximum AIM classification recognize 36 bits instead of the original 18 bits.

dial_up_.pl1

This program handles all logging in to the system. Changes had to be made to allow users coming in via the Internet to login. The change was needed to make the login channel allow entry at classifications above UNCLASSIFIED. At a later date, another change was made to allow users to create a new process at a different AIM classification without logging out and back in again.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the following people for their contributions to the modifications of the Multics AIM mechanism for DOCKMASTER. Their efforts were essential to the development, programming, testing, and review of these changes.

James Arnold, National Computer Security Center
T. Allen Grider, Honeywell Federal Systems, Inc.
John Rutemiller, National Computer Security Center
Grant Wagner, National Computer Security Center
John Wyszynski, National Computer Security Center

The authors also wish to the acknowledge the following people for their invaluable input in developing and refining this paper.

Diann Carpenter, National Computer Security Center
William Geer, National Computer Security Center
Jack Holleran, National Computer Security Center
Grant Wagner, National Computer Security Center

BUILDING TRUST INTO A MULTILEVEL FILE SYSTEM

Cynthia E. Irvine, Todd B. Acheson, and Michael F. Thompson
Gemini Computers, Inc.
2511 Garden Road
Monterey, California 93940

Abstract

File systems are an intrinsic part of any operating system providing support for a general application environment. To help provide general operating system functionality, a multilevel file system is being built to run on the GEMSOS TCB. The process of designing a file system for a multilevel environment, although similar in many respects to that for its untrusted counterpart, should include consideration of factors which will render its structure consistent with the trusted environment upon which it is built. The file system should take advantage of the security mechanisms available from the TCB.

In this paper, two techniques are described which contribute to building trust into a file system design. The first is the use of mandatory access controls as a constraining design guide, and the second is the use of the intended discretionary access control policy as a driver for design choices.

1. Introduction

At Gemini, significant effort is being focused on the development of general purpose operating system support to execute as an untrusted application on a Trusted Computing Base (TCB) having the highest level of assurance. The underlying TCB is the Gemini Multiprocessing Secure Operating System (GEMSOS) [1], which is targeted for evaluation for a Class A1 rating according to the Trusted Computer System Evaluation Criteria (TCSEC) [2]. A requirement for an A1 TCB is the exclusion of non-security relevant functionality from the TCB. Thus it is the operating system executing on the TCB which will provide the usual range of capabilities: memory management services, process management, I/O device management, and file system services. The purpose of this paper is to examine how one of these services, a multilevel file system, can be designed to utilize the trust that has been achieved in the security mechanisms of the underlying high assurance TCB. First, environments of the highest assurance and those of lower assurance are contrasted and the overall impact of high assurance constraints on file system design is described. Then, approaches to file system design consistent with requirements for the highest levels of assurance are considered from the perspective of both Mandatory Access Controls (MAC) and Discretionary Access Controls (DAC). Design techniques for the overlying multilevel environment are described and, for developers of low assurance systems, methods are discussed which may be employed to avoid intrinsic security flaws. The effect of security policy on file system design is reviewed.

2. File System Design for the Highest Levels of Assurance

The requirements of the TCSEC for assurance undergo a major transition between Class B2 and Class B3. It has been argued [3] that only systems of Classes B3 and A1 can be categorized as high assurance because only systems in these evaluation classes are able to conclusively demonstrate that they contain a security kernel. Systems of Classes B2 and lower cannot make this claim. To better address the issues encountered when developing a file system for use on a high assurance "environment," it is helpful to examine how file system support may differ between high and low assurance trusted systems.

Architectural requirements at Class B2 and below permit a the TCB perimeter to encompass an entire general purpose operating system, of which a major component is likely to be file management services. Within the TCB perimeter, it is possible for selected, or perhaps a large number of, operating system functions to be multilevel. As such, these systems are able to manipulate complex multilevel data structures of file system resources in support of single-level untrusted application code [4].

General purpose file system portions of an operating system are fundamentally not protection-critical. Historically this claim has been recognized and validated in that, for high assurance systems, the file systems have been constructed outside of the TCB. Early file systems exhibiting this choice were the Secure UNIX System described for KSOS [5]; the Project Guardian design [6] in which layering separated mandatory access controls from the rémaining operating system services; the file system designed for SCOMP [7]; and the Mitre security kernel design [8]. More recently, the need to separate security-relevant from other functionality has been reiterated [3].

A TCB must include the "totality of protection mechanisms ... the combination of which is responsible for enforcing a security policy" [2]. From the perspective of systems at lower levels of assurance one might attempt to argue that file systems are indeed protection critical, however, the existence of worked examples in high assurance systems of several file system designs in which the file systems are outside of the TCB demonstrates that this criticality may be attributed to particular file system choices rather than intrinsic file system requirements.

At Class B3, the introduction of the architectural requirement of minimization of the TCB to include only security relevant functionality results in a fundamentally different approach to system development as contrasted with that permissible at lower assurance levels. TCBs of the highest assurance are required by the TCSEC to use significant system engineering directed toward minimizing the complexity of the TCB, excluding from it modules that are not protection-critical. Thus the TCSEC builds on historical precedent. It follows that a general purpose file system cannot necessarily be incorporated wholesale into a high assurance TCB. Any data structures that contain information over a range of access classes potentially require management by a multilevel or trusted subject. If used, multilevel subjects are inherently part of the TCB, but we suggest that their use for the manipulation of the complex data structures required for a file system would tremendously complicate the verification and successful evaluation of a TCB at the highest evaluation classes. Thus current thinking of trusted system design in a high assurance environment indicates that a general purpose file system must be provided by single level operating system subjects effectively executing as "applications" on the TCB.

3. Approaches to File Systems in High Assurance Environments

An objective of multilevel security is to allow subjects having different access classes to execute on the same system while prohibiting information from flowing "down." Two approaches to file management services have been identified for high assurance multilevel environments: first, multiple, isolated single-level file systems providing complete compatibility with a non-secure system interface from within each single level, and, second, a multilevel file system which overcomes this isolation to provide many features of the untrusted general purpose operating system but which, due to the constraints of high assurance, may sacrifice some compatibility with a comparable non-secure system interface.

The first is the virtual machine approach: the untrusted operating system executes on a single level virtual machine provided by the TCB [9]. Having negotiated a particular session level, the operating system executes as a single level subject in an isolated environment. In its simplest form, the virtual machine mode yields a complete single level environment and no access to information at different sensitivity levels is possible. This highlights one of the primary advantages of the virtual machine approach: it may be relatively easy to implement because it may utilize an unmodified preexisting operating system. If this operating system supports a rich set of applications, its second advantage becomes clear: the virtual machine can support a large body of existing application software. A drawback of the virtual machine approach is its per-access class isolation of file system data structures.

If a "multilevel" file system is to be provided, there are significant operational consequences of the virtual machine approach. First, subjects with a high sensitivity level may need to access low and high sensitivity information simultaneously and on a routine basis. Although examples of extended virtual machine approaches, such as the secure DOS demonstration on GEMSOS [10], exist in which subjects at higher access classes are provided with read access to objects at lower access classes, in general, this is difficult because the virtual machine monitor would be required to manage access to the multilevel file system and thus would be invested with substantially increased complexity and hence would be noncompliant with the TCSEC at the highest evaluation classes. To support the ability of high access class subjects to "read down," considerable additional functionality may be required within the file management services. The file system interface must provide a means for applications to specify which file in which file system is to be accessed. At this point, the advantage of non-customized software has been lost.

For the virtual machine approach, a mechanism is needed for the use of file systems at each supported access class. Before a user selects a new session level in which files are to be created, a minimal file system data structure at that level must be present. If construction is postponed until just prior to the user's instantiation at a particular session level, then a multilevel subject or an administrator must be provided to deal with the differing access classes. Alternatively, if some minimal file system is to be provided when the system is first initialized, then a substantial waste of system resources may result. To illustrate this last point, consider the number of access classes possible using the GEMSOS TCB: with two sets of 16 hierarchical levels and a total of 96 non-hierarchical categories, the potential number of access classes, although enumerable, is extremely large. To create even the most minimal file system objects at such a large number of access classes when it is expected that most will never be used during the entire lifetime of the system would be extravagantly wasteful, if it were possible.

Despite additions to provide an apparent multilevel file system in the virtual machine monitor milieu, some do not regard such designs to be truly multilevel file systems.

An alternative approach, a coherent multilevel file system, is in greater harmony with the spirit of multilevel security. In essence, the file system contains objects at many different access classes, and is designed to provide subjects at different access classes with a unified view of the file system while the TCB constrains each subject to access only those file system objects consistent with the policy being enforced by the underlying TCB. This approach allows the effective use of files in a coherent multilevel environment.

4. Using MAC to Build Trust

By relying on the security mechanisms provided by an underlying high assurance TCB, a file system can be designed which, in effect, extends the TCB's trustworthiness to the file system interface. In a low assurance system, the quest for compatibility with non-multilevel general purpose file systems may lead to certain problems which at high assurance cannot even be contemplated due to the fact that the file system data structure is managed by untrusted single level subjects. In the following sections, several of these design issues will be examined and it will be shown that by assuming that the file system is built on an underlying high assurance TCB, it is possible to avoid certain classes of potential problems and, perhaps, to achieve a more robust design with respect to trust.

4.1 Multilevel Tables

In the UNIX [11] operating system, the focus of the file system mechanism is the inode. It contains the attributes of the file system object as well as the address of the object's data. In any physical file system, where by "physical" we mean the partitioning of some storage device or devices into logical mountable file system volumes, each inode is distinct. Clearly, the notion of a multilevel inode table is a possibility for low assurance systems where the complex file system mechanism is inside the TCB. Even in this case the multilevel inode table introduces an information channel as high- and low-level applications request the allocation of inodes. In contrast, a file system built to meet high assurance architectural requirements cannot use a multilevel inode table because the file system will be managed by single level subjects which are unable to manipulate objects over a range of access classes. We propose that by taking the time to consider high assurance engineering issues, the builders of low assurance systems can decrease the likelihood of introducing channels.

4.2 Multilevel Directories

For low assurance systems in which subjects managing the file system are multilevel, the use of multilevel directories may seem appealing at first. Here a directory would be a labeled object, but would contain entries each of which would have a separate label. Unfortunately, this leads to the requirement that the label on the directory as a whole be that of its most sensitive entry. By recursion, it can be seen that the root directory of the file system would assume the access class of the most sensitive entry in the file system. Adding another more sensitive entry would result in massive relabeling, quite to the contrary of accepted notions of tranquility [12].

4.3 Upgraded File System Objects

One can postulate a hierarchical file system in which any file system element may be at a higher access class than its parent, i.e., upgraded, thus implying that any directory might contain upgraded files, directories, and links. Here we examine the implications of circumscribing this unlimited flexibility.

The primary difficulty with upgraded objects of all types is that of object deletion (a topic which also will be discussed in a different context in the following section). To delete an upgraded object, a "range" is required because it is not only necessary to delete the high access class object, a write at the high access class, but also to delete the entry for that object in the lower access class parent object, a write at the lower access class. Deletion is not a problem at low assurance where the management of file services is within the TCB. Because of the inherent trust that must be placed in a file system mechanism of this nature, it is able to delete objects of upgraded access classes which are associated with objects at lower access classes. For a file management system executing at a single level outside of a high assurance TCB, such file deletion is impossible. A single level subject cannot delete the upgraded object and also modify the lower access class parent so that its entry in the parent object is removed. A trusted subject is required, e.g., as part of a security administrator's special tools. In order to minimize potential operational difficulties associated with invoking a trusted subject for routine file system management, we recommend that only directories be upgraded objects. Invoking the arguments for what has been termed "compatibility" [12], we also recommend that all directories dominate the access classes of their parent directories.

Operationally this makes sense. Within any directory all of the files would be at the same sensitivity level. The level of a directory would indicate the intent of the creator to populate the directory with files at that level. Similar files at other levels would occupy other directories. Given that each file, which is simply a data repository will match the level of its directory, the directories themselves are the candidates to be the upgraded objects which can allow the file system to be multilevel. If users create their major directories on the basis of access class, then the use of the special services required to delete an upgraded directory will be infrequent.

4.4 Deletion of File System Objects

In some file systems, e.g., UNIX, if a file is being simultaneously accessed by one or more subjects and some other subject chooses to delete the file, the space occupied by the file is not released for reallocation until all subjects terminate their access to the file [13]. Thus, as a result of its global knowledge of current accesses to the file, the operating system postpones the act of file deletion.

The constraints imposed by an underlying high assurance TCB on single level subjects both accessing and manipulating the file system result in behavior dissimilar to that possible at lower levels of assurance. Single level operating system low access class subjects cannot have global knowledge of file usage at higher access classes: they must have no way of detecting that low access class files are in use by subjects at higher access classes, otherwise an information flow from high to low level subjects would exist, viz., the TCB would contain a design flaw. While at any access class the operating system subject may have read access to global databases reflecting current accesses to a file at the subject's current level and below, mandatory access policy will make it impossible for an untrusted subject to detect file system accesses by subjects at higher levels. This means the operating system acting on behalf of a subject at the access class of the file is obliged to carry through to completion any command for deletion that is both permitted under DAC and not blocked by active accesses of subjects at the same access class.

Applications will have to cope with the deletion semantics. "Secure" applications, designed to use a multilevel file system, can make special checks preceding or following attempts to obtain information from deletable objects. If a "secure" application running at a higher access class has a temporary copy of the deleted object, it can make a copy in its own directories at the higher access class. The behavior of old non-secure applications is more problematic: if executing at a high session level, they may or may not return meaningful error messages and take appropriate actions accordingly when files at lower access classes are deleted. Some old applications, if mindlessly introduced into a multilevel environment, may be expected to fail with consequences which from the user's perspective may be considered serious.

4.5 Links: Hard Versus Symbolic

Links present another area where the behavior of software may be significantly modified in a multilevel file system implemented on an underlying high assurance TCB. The objective of links in both the trusted and untrusted worlds is to allow users to share files. In a trusted environment, a desirable scenario might be to allow links from a directory at one access class to objects in other directories which are at either higher or lower access classes. Despite the fact that, in a low assurance system, links may be managed within the TCB, it is useful to examine the impact of a high assurance environment on link management. There are two kinds of links: hard links and symbolic links.

Hard links are found in the single-level AT&T version of the UNIX operating system [12, 13]. Links are from the directory to the UNIX inode. Within the inode a "link count" is maintained so that only when the last link is deleted does the inode disappear. UNIX-like links traditionally are not allowed to cross physical file system boundaries. The principle advantage of hard links is compatibility with the way interactive users construct their file environments and, to a lesser extent, with preexisting software. Unfortunately, there are several severe drawbacks to using hard links in a multilevel environment. Because it would be necessary for subjects at a variety of access classes to modify link counts, a single, multilevel file system cannot be implemented at Class B3 or above using pure hard links without conceding the existence of high bandwidth covert storage channels. In reality, a high assurance TCB will not allow subjects at differing access classes to modify link counts, thus relegating hard links to single level semantics. In addition, the use of hard links in combination with mechanisms such as mountable file systems where each file system is single level and at a different access class results in difficult file sharing at different levels because hard links cannot cross file systems.

Symbolic links find one of their best examples in the Multics system [15]. Their semantics are well understood and complementary to multilevel security [16]. A similar mechanism has been introduced more recently into the BSD versions of UNIX [17]. A symbolic link is from a directory to a symbolic name, where the symbolic name may be any of several file system object or data structures. If the target object is deleted, the symbolic link remains unchanged: only when an attempt is made to access the deleted object, does its deletion become apparent. If the original object is deleted and another object having the same name is substituted in its place, then access via the symbolic link will result in the access to the new object. Symbolic links may be used to cross physical file systems and are thus well suited to a multilevel environment.

At first glance, the use of symbolic links which point to objects at higher or lower access classes would seem to make the directories containing them multilevel objects, but this is not the case. (It is sufficient to note, as pointed out earlier, that one might "get into trouble" with the notion of directories as repositories for information at many access classes since the level of the directory would necessarily be that of the most sensitive object it contained, and, by recursion, the root directory of the file system would be forced to assume the access class of the most sensitive object in the file system.) On the contrary, symbolic links contain no information about the object to which they point. The link is at the access class of the directory in which it is contained even though a link to a higher access class object typically is intended for use by a subject at a higher access class. Conversely, since theoretically a subject at a higher access class can observe the path to an object at a lower access class, that subject can easily create symbolic links to objects at lower access classes.

4.6 The Location of Temporary Files

To be most attractive, a multilevel operating system should support a large body of existing applications software including software development tools such as editors, compilers, archivers, etc. Many of these tools require the use of temporary files located in a common directory. There are two reasons why directories for temporary files which are shared across access classes present a problem in a multilevel file system. The first is the potential for duplicate names as subjects at differing access classes create temporary files in the same directory. In order to handle duplicate names, software would have to use "access class" as an additional qualifier for the file name. Since that might entail modification of applications and the objective is to use off-the-shelf software, this is not practical. The second reason is the problem of deleting upgraded objects from the temporary directory. To allow access to a single temporary directory by subjects at all access classes, that directory would have to be system low. Thus subjects at higher access classes would have to create upgraded entries in the system low directory. The security policy requires a subject with a range encompassing that of the directory and the temporary file (viz., trusted) to delete upgraded objects. This is possible in low assurance systems where the file system is part of the TCB; it is precluded from the options available on high assurance systems where the file system must be excluded from the TCB.

A solution to both the duplicate name and the deletion problems is to create a special directory type for temporary directories akin to the hiding directories found in the Linus IV [18] system. Below the visible temporary directory there exists a hidden subtree of directories at each access class. All accesses to temporary files by subjects at a particular access class are targeted by the operating system to the hidden subdirectory at the corresponding access class. Name duplication at differing access classes is no longer a problem because the operating system implicitly extends the name of each temporary file with the access class of the subject creating it. Since the temporary files are at the same access class as their containing directory, the deletion problem is resolved. This mechanism is hidden from applications. Because the operating system does the work by recognizing the paths to temporary directories as special, applications can be used unmodified. In the case of GEMSOS, a multilevel subject within the TCB creates a location at each new access class which can be used by the untrusted operating system as a foundation upon which a temporary directory at that access class can be built.

The examples of this section illustrate that file system designers constrained by an underlying high assurance TCB do not have the spectrum of choices, some of which are ill advised, available to file system designers in low assurance environments where the file system and the TCB can overlap.

5. Using DAC to Build Trust in a Coherent Multilevel File System

In a high assurance system a DAC policy is enforced at the TCB interface. The operating system, built on the underlying TCB, has the opportunity to maximize its usage of the underlying DAC mechanisms in order to enhance its trustedness.

DAC, discretionary by definition, can be rendered ineffective by allowing untrusted applications (in this case the operating system itself) to give away the protection features provided by the TCB. A file system poorly designed with respect to trust can fail to reflect the intended DAC policy at its interface, but instead may implement a mechanism for which the policy is ill-defined or even flawed. During the course of our work at Gemini, several seemingly elegant file system designs have been found to be flawed with respect DAC. Their common trait was the need for global data structures for which all operating system subjects required DAC access.

It is possible to arrange the file system such that the objects used to contain its access control lists (ACLs) have a flat structure and are used much like UNIX inode tables [13] to construct the observable files and directories. However, in order to set the ACLs for file system objects, all subjects must have DAC permission to modify the objects containing the ACLs. Many such designs also required other global data structures with unrestricted DAC access granted to all subjects. Thus it follows that a penetrator of the operating system (not the TCB) could use the existing operating system software (no operating system Trojan Horses are needed) to obtain unauthorized access to information. In effect, the operating system in such a case has been designed so that it must assume much of the responsibility for the enforcement of DAC policy. Having implemented such design choices, one is not able to "trust" the file system with even close to the same degree of assurance as the DAC policy enforced at the TCB interface.

In contrast, by giving careful attention to the constraints of the multilevel environment, the file system may be designed to be constrained by, and reflect the DAC enforced by, the TCB. In this case, operating system penetrators are foiled by the combined structural properties of the file system and TCB-enforced DAC.. In very simple terms, the file system is built without using global data structures for the propagation of DAC. An ACL is associated with each file system object used to contain directory and file status information, and directory and file data. A component of a directory would be the repository for the ACLs associated with the files and subdirectories which are direct entries in that directory. Thus, the access to an ACL does not require access to a structure to which all subjects have DAC access. DAC may be applied individually to each component of the file system. Users are able to walk the file system and list directories without necessarily having access to the files located "in" the directories, or, conversely, they may have access to the files without being granted access to the directories. Directories and similar structures may be arranged to promote efficient file system transit, while files may be designed to provide for efficient data access. Details of such a design are intended for a future presentation.

By synergistically combining TCB-enforced DAC and operating system structures, it is possible for TCB-enforced DAC to be mapped directly to the file system, or, more broadly, the operating system interface. This strong linkage between TCB-enforced DAC and that observed by applications ensures "trust" in the file system in a practical and operational sense.

6. Distinguishing Security Policy from Common Sense

When considering a file system for use in a high-assurance trusted environment, it is essential to distinguish between features that are reflections of the security policy and those that are related to functionality. Security policy pertains exclusively to the access of users to information where secrecy policies protect information from unauthorized disclosure and integrity policies protect information from unauthorized modification. Non-security relevant mechanisms established a system-level etiquette for general operations. Sometimes it is difficult to differentiate security and non-security relevant functionality.

Consider the access to the time that a file system object has been modified. Some implementations may allow this access to take place despite lack of access to the object itself or to its containing directory, while others may severely constrain it. A mechanism allowing or disallowing access to time modified is one related to system functionality policies not to security policy. It is clear, however, that if the system encompasses a mechanism requiring that all users, regardless of their session levels, be allowed access to all modification times, regardless of the access class of the object in which the times are stored, this would constitute a serious security flaw because the times could be used as a covert channel.

Another example is the "execute" access used on UNIX directories and other modes of access contemplated for UNIX-like systems [19]. In this case, there has been considerable confusion as to their relevance in security policy enforcement and their real purpose, which is to provide other desirable properties that may not be inherently security relevant. The "execute" permission to directories allows users access to files within a directory, where for each file the user must have the appropriate form of access on the file itself. To "read" the contents of a directory, a user must explicitly be given "read" permission to the directory. As repositories for data, access to files and directories is mediated by Mandatory Access Controls and by permissions granted in ACLs. The notion of accessing a file irrespective of the permissions associated with the information in the containing directory is supportable and an example already exists with Multics [15]. The "execute" permission for directories makes explicit that which is implicit in some existing systems. It does not control disclosure of information in the directory and we conclude that it is a functional rather than a security feature.

Other features of the file system that are engineering choices driven by functional requirements rather than security requirements include maintenance by UNIX [SVID] of the date of last use and the date of last modification of status of a file or directory (the latter is the time at which either the time of last access or time of last modification were modified). Both are relevant from the standpoint of useful data for file and directory management; however, neither are relevant to the enforcement of an access control security policy. Obviously, where a high assurance TCB provides the foundation for an operating system populated by single level subjects, neither of these two file system object attributes can be maintained for all accesses to files and directories by all single level subjects. They are only maintained in objects at and modifiable by subjects at the access class of the file or directory. Such choices lead to nominal incompatibilities with file systems conceived outside of the multilevel universe. These incompatibilities are inevitable, but are usually of acceptable impact since, in this case, there are no applications expecting such multilevel support.

7. Summary

It has been shown that building trust into a coherent, multilevel file system requires careful design with respect to untrusted components. It is necessary to make a consistent set of choices which clearly define what the behavior of the secure system will be. These include the use of symbolic rather than hard links, the design of applications to avoid the dangers of deleted objects, the creation of special mechanisms to handle temporary directories, and the restriction of upgraded objects to directories.

This paper has presented concepts that are useful in constructing data structures for use in a high assurance, multilevel environment. Issues related to the design of file management services are presented. We argue that to build "trust" into a file system, the designer's objective should be to take "trust" out of the file system and to "trust" the underlying TCB instead. Design choices to build trust into a file system can be made with respect to both MAC and DAC. To build trust using MAC, one should always constrain the design to meet the architectural and engineering requirements of the highest possible TCSEC evaluation class, viz., Classes B3 and A1. To build trust using DAC, the design should not subvert the high assurance DAC supported by underlying access control mechanisms, but instead should attempt to provide a strong structural linkage between the underlying DAC enforcement mechanisms and the DAC presented at the file system interface.

The authors wish to thank Mark Heckman and Timothy Levin for their careful review and thoughtful comments on this paper.

References

[1] Schell, R.R., Tao, T.F., and Heckman, M. "Designing the GEMSOS Security Kernel for Security and Performance." in *Proc. 8th National Computer Security Conference*, pp. 108-119, 1985.

[2] ----, *Department of Defense Trusted System Evaluation Criteria.* DOD 5200.28-STD, National Computer Security Center, Fort Meade, MD, 1985.

[3] Shockley, W.R., Schell, R.R., and Thompson, M.F., "The Importance of High Assurance Computers for Command, Control, Communications, and Intelligence Systems." in *Proc. of the Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, pp. 331-341, 1987.

[4] Gligor, V. D., Burch, E. L., Chandersekaran, C.S., Chapman, R.S., Dotterer, L.J., Hecht, M.S., Jiang, W.E., Luckenbaugh, G.L., and Vasudevan, N., "On the Design and the Implementation of Secure Xenix Workstations," in *Proc. IEEE Symp. on Security and Privacy*, pp. 102-117, 1986.

[5] ----, "Secure Minicomputer Operating System (KSOS) Final Report, Department of Defense Kernelized Secure Operating System." Ford Aerospace and Communications Corporation, Palo Alto, CA, 1977.

[6] Schroeder, M.D., Clark, D.D., and Saltzer, J.H., "The Multics Kernel Design Project." *Proc. Sixth ACM Symp. on Operating System Principles*, p. 43, November 1977.

[7] Fraim, L.J. "SCOMP: A Solution to the Multilevel Security Problem". *Computer*, 16, No 7, pp. 26-34, 1983.

[8] Schiller, W.L., "The Design and Specification of a Security Kernel for the PDP-11/45." ESD-TR-75-69. Hanscom AFB, MA. AFESD, 1975.

[9] Gold, B.D., Linde, R.R., Peller, R.J., Schaefer, M., Scheid, J.F., and Ward, P.D. "A Security Retrofit of VM/370." *Proc. of the NCC*, 48, p. 335, 1979.

[10] Shockley, W.R., Tao, T.F., and Thompson, M.F. "An Overview of the GEMSOS Class A1 Techonology and Application Experience." in *Proc. of the 11th National Computer Security Conference*, p. 238-245, 1988.

[11] ----, *System V Interface Definition*, Volumes I and II. AT&T. Indianapolis, Indiana, 1986.

[12] Bell, D.E., and LaPadula, L.J. "Computer Security Model: Unified Exposition and Multics Interpretation." Tech. report ESC-TR-75-306, MTR-2997 Rev. 1, The Mitre Corporation, Bedford, MA, 1976.

[13] Bach, M.J. *The Design of the UNIX Operating System.* Prentice Hall, Englewood Cliffs, NJ, p. 61, 1986.

[14] Denning, D.E., Lunt, T.F., Schell, R.R., Heckman, M., and Shockley, W. "A Multilevel Relational Data Model." in *Proc. IEEE Symp. on Security and Privacy*, p. 220, 1987.

[15] Organick, E. I., *The Multics System: An Examiniation of Its Structure.* MIT Press, Cambridge, MA, p. 222, 1972.

[16] Whitmore, J., Bensoussan, A., Green, P., Hunt, D., Kobziar, A., and Stern, J. "Design for Multics Security Enhancements." ESD-TR-74-176, Hanscom AFB, MA.

[17] McKusick, M. K., Joy, W. N., Leffler, S. J., and Fabry, R. S., "A Fast File System for Unix." *ACM Transactions on Computer Systems*, Vol. 2., No. 3, p. 181, August 1984.

[18] Kramer, S. "Linus IV - An Experiment in Computer Security." in *Proc. IEEE Symp. on Security and Privacy*, pp. 24-32, April 1984.

[19] ----, Trusted UNIX Working Group (TRUSIX) Rationale for Selecting Access Control List Features for the UNIX System. NCSC-TG-020-A. Version 1. National Computer Security Center, Fort Meade, MD, 1989.

LAVA/CIS Version 2.0:

A SOFTWARE SYSTEM FOR VULNERABILITY AND RISK ASSESSMENT

S. T. Smith and M. L. Jalbert
Safeguards Systems Group, MS-E541
Los Alamos National Laboratory
P. O. Box 1663
Los Alamos, New Mexico 87545

ABSTRACT

LAVA (the Los Alamos Vulnerability/Risk Assessment system) is an original systematic approach to risk assessment developed at the Los Alamos National Laboratory It is an alternative to existing quantitative methods, providing an approach that is both objective and subjective, and producing results that are both quantitative and qualitative. LAVA was developed as a tool to help satisfy federal requirements for periodic vulnerability and risk assessments of a variety of systems and to satisfy the resulting need for an inexpensive, reusable, automated risk assessment tool firmly rooted in science. LAVA is a three-part systematic approach to risk assessment that can be used to model a variety of application systems such as computer security systems, communications security systems, information security systems, and others. The first part of LAVA is the mathematical model based on classical risk assessment, hierarchical multilevel system theory, decision theory, fuzzy possibility theory, expert system theory, utility theory, and cognitive science. The second part is the implementation of the mathematical risk model as a general software engine executed on a large class of personal computers. The third part is the application data sets written for a specific application system. The user of a LAVA application is not required to have knowledge of formal risk assessment techniques. All the technical expertise and specialized knowledge are built into the software engine and the application system itself. LAVA application systems, including the popular computer security application, have been in use by federal government agencies since 1984; the previous computer security version–LAVA/CIS, Version 1.01 [34]–is used by over 100 agencies at more than 500 sites.

INTRODUCTION

LAVA (the Los Alamos Vulnerability/Risk Assessment system) is an original systematic approach to risk assessment developed at the Los Alamos National Laboratory to determine vulnerabilities and risks inherent in massive, complicated systems. Characteristics of such systems are huge bodies of imprecise data, indeterminate (and possibly undetected) events, large quantities of subjective information, and a dearth of objective information. LAVA was developed as a tool to help satisfy federal requirements for periodic vulnerability and risk assessments of a variety of systems and to satisfy the resulting need for an inexpensive, reusable, automated risk assessment tool firmly rooted in science [1]. When the LAVA project began in 1983, there was no such tool [2]; LAVA was designed to fill that gap [3].

LAVA is an alternative to existing quantitative methods, providing an approach that is both objective and subjective, and producing results that are both quantitative and qualitative. In addition, LAVA is used by some agencies as a self-testing aid in preparing for inspections, as a self-evaluating device in testing compliance with the various orders and criteria that exist, and as a certification device by an inspection team.

LAVA is a three-part systematic approach to risk assessment that can be used to model a variety of application systems such as computer security systems, communications security systems, information security systems, and others. The first part of LAVA is the mathematical model based on classical risk assessment [4,5], hierarchical multilevel systems theory [6,7], decision theory [8-10], fuzzy possibility theory [11-15], expert system theory [14,15], utility theory [17,18], and cognitive science [19,20]. (The mathematical model has been presented at other technical meetings [21-23], and generally will not be addressed in depth in this paper.) The second part is the implementation of the mathematical risk model as a general software engine, an expert system framework written in a commercially available programming language for a large class of personal computers. The third part comprises the application data sets written for a specific application system; each application system is a knowledge-based expert system. LAVA provides a framework [24] for creating applications upon which the software engine operates; all application-specific information appears as data.

The user of a LAVA application is not required to have knowledge of formal risk assessment techniques. All the technical expertise and specialized knowledge are built into the software engine and the application system itself. LAVA applications include the popular computer security application [27-30] and applications for nuclear power plant control rooms [31], embedded systems, survivability systems, transborder data flow systems [32], and property control systems. We presently are developing LAVA applications for nuclear processing plant safeguards systems [33] and operations security systems and are discussing the development of a LAVA application for environment, health, and safety issues. LAVA application systems have been in use by federal government agencies since 1984; the previous version–LAVA/CIS, Version 1.01 [34]–is used by over 100 agencies at more than 500 sites.

LAVA/CIS: THE COMPUTER/INFORMATION SECURITY MODEL

The LAVA system was used to develop a hierarchical structure and sets of fuzzy analysis trees for modeling risk assessment for systems associated with computer and information security. Knowledge-based expert systems were built with LAVA to assess risks in application systems comprising a subject system and a safeguards system. The subject system model is sets of threats, assets, and undesirable outcomes; because the threat to security systems is ever-changing, LAVA includes a dynamic threat analysis [25,26]. The safeguards system model has three parts: sets of safeguards functions for protecting the assets from the threats by preventing or ameliorating the undesirable outcomes that may happen to the assets, sets of safeguards subfunctions whose performance determine whether the function is adequate and complete, and sets of issues, appearing in the software as interactive questionnaires, whose measures (in both monetary and linguistic terms) define both the weaknesses in the safeguards system and the potential costs of an undesirable outcome occurring as a result of a successful attack against safeguards system weaknesses.

For the computer/information security application model, LAVA/CIS, the set of postulated assets consists of four categories: (1) the facility, including physical plant and personnel; (2) hardware, including all computing and ancillary pre- and postprocessing hardware; (3) machine-interpretable information, including software, input and output files, and databases; and (4) human-interpretable information, including documents, screen displays, graphs, charts, film output, and so forth. The model's threat set consists of three categories: 1) natural, random, and environmental hazards; 2) direct or onsite humans, including the authorized insider; and 3) indirect or offsite humans (but this threat category has not yet been implemented in the software). Figures 1 and 2 show the hierarchical structures for the three threat categories with respect to the four asset categories. Included as the third and fourth levels in these hierarchies, and discussed later in this paper, are representative safeguards functions and subfunctions associated with each threat-asset pair; Fig. 3 shows the complete analysis structure for the [direct human threat, software asset] combination.

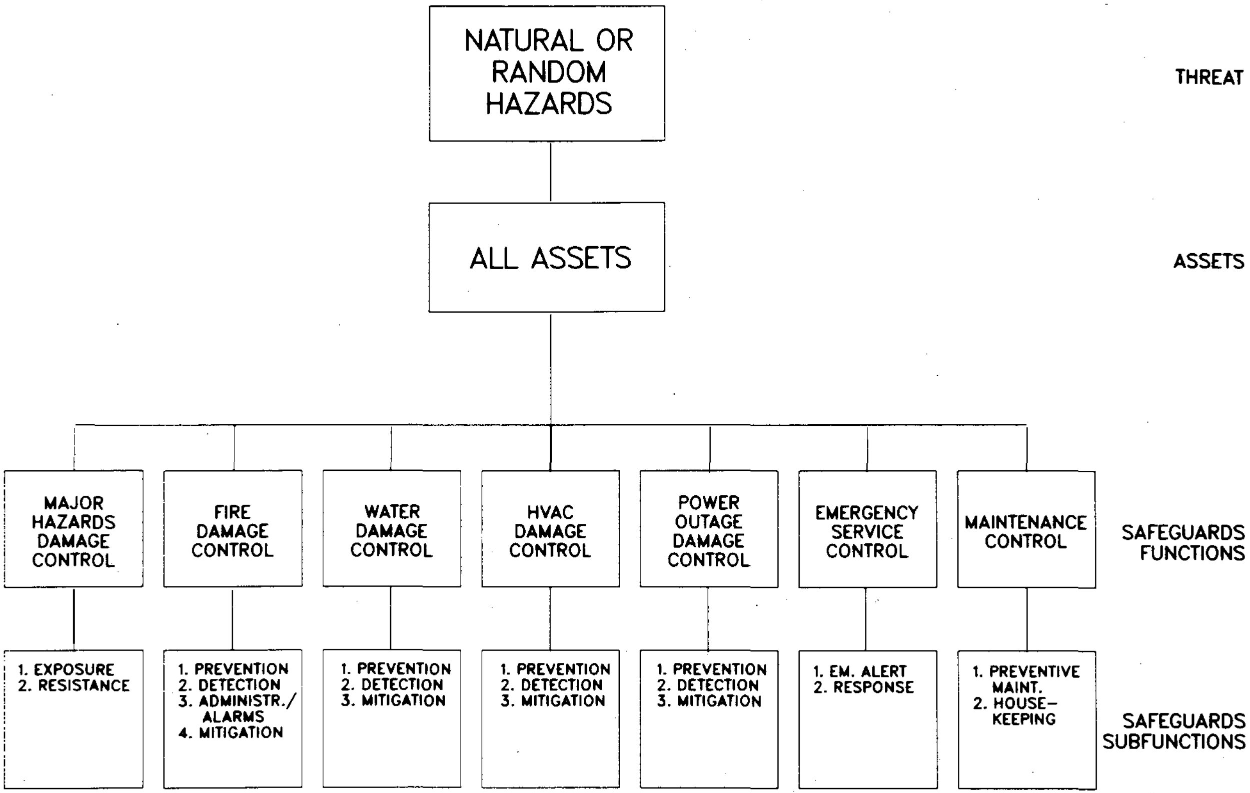
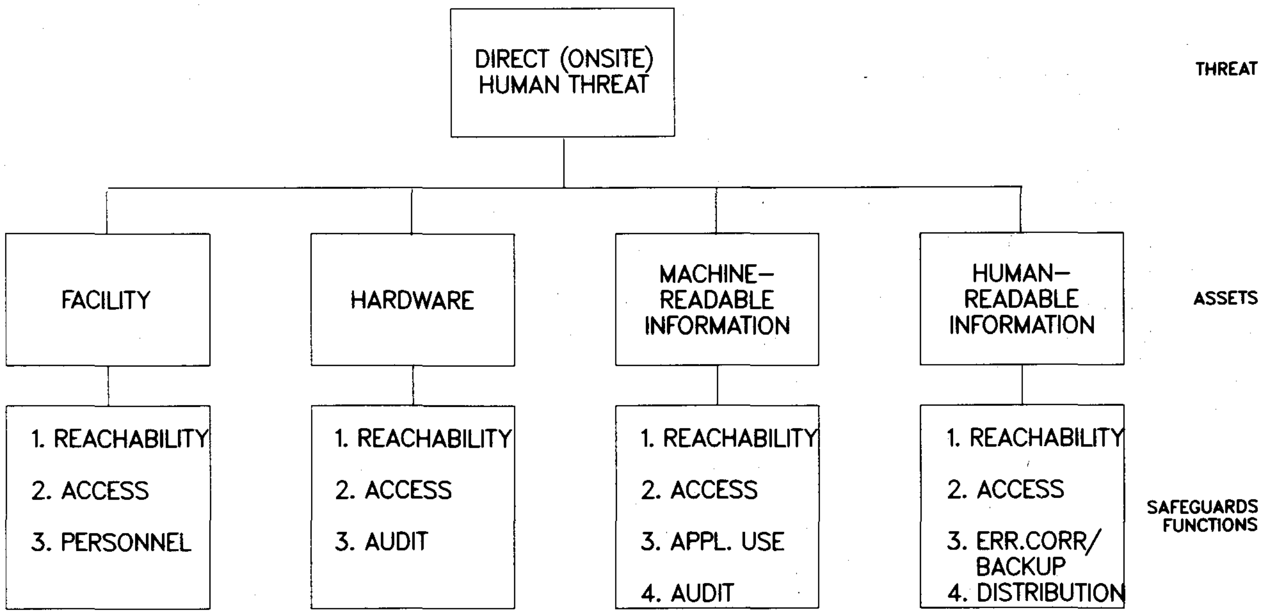

NATURAL OR RANDOM HAZARDS
THREAT
ALL ASSETS
ASSETS
MAJOR HAZARDS DAMAGE CONTROL
FIRE DAMAGE CONTROL
WATER DAMAGE CONTROL
HVAC DAMAGE CONTROL
POWER OUTAGE DAMAGE CONTROL
EMERGENCY SERVICE CONTROL
MAINTENANCE CONTROL
SAFEGUARDS FUNCTIONS
1. EXPOSURE 2. RESISTANCE
1. PREVENTION 2. DETECTION 3. ADMINISTR./ ALARMS 4. MITIGATION
1. PREVENTION 2. DETECTION 3. MITIGATION
1. PREVENTION 2. DETECTION 3. MITIGATION
1. PREVENTION 2. DETECTION 3. MITIGATION
1. EM. ALERT 2. RESPONSE
1. PREVENTIVE MAINT. 2. HOUSE-KEEPING
SAFEGUARDS SUBFUNCTIONS

Fig. 1. Natural Hazards Hierarchy for Computer/Information Security Application


DIRECT (ONSITE) HUMAN THREAT
THREAT
FACILITY
HARDWARE
MACHINE-READABLE INFORMATION
HUMAN-READABLE INFORMATION
ASSETS
1. REACHABILITY 2. ACCESS 3. PERSONNEL
1. REACHABILITY 2. ACCESS 3. AUDIT
1. REACHABILITY 2. ACCESS 3. APPL. USE 4. AUDIT
1. REACHABILITY 2. ACCESS 3. ERR.CORR/ BACKUP 4. DISTRIBUTION
SAFEGUARDS FUNCTIONS

SAFEGUARDS SUBFUNCTIONS BRANCH FROM EACH SAFEGUARDS FUNCTION.

Fig. 2. Direct (Onsite) Human Threat Hierarchy for Computer/Information Security Application

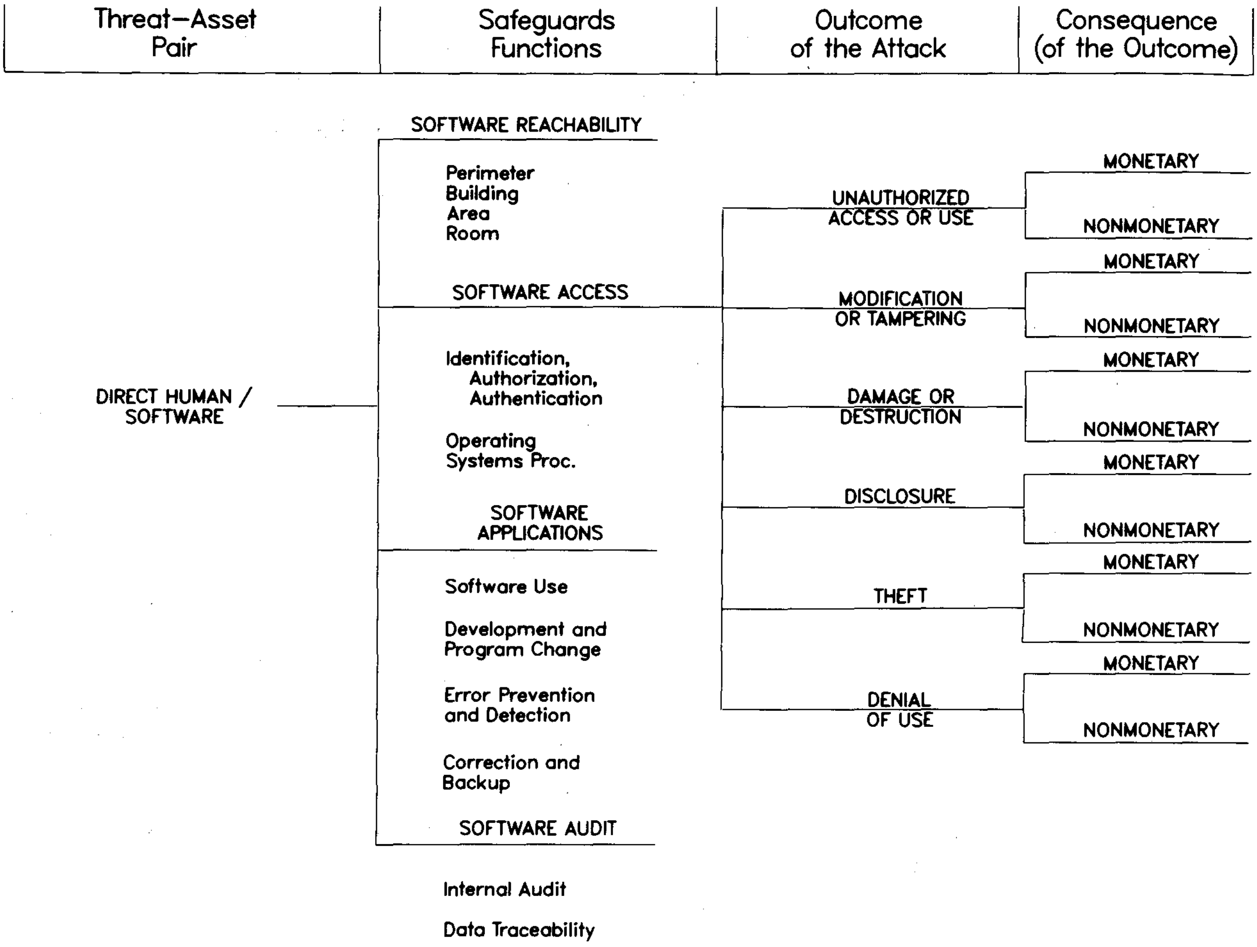


Fig. 3. Direct Human-Software Scenario Analysis Tree for Computer/InformationSecurity Application

Six undesirable outcomes are considered in the computer/information security model: (1) unauthorized access or use; (2) damage, modification, or tampering; (3) destruction; (4) theft; (5) unauthorized disclosure; and (6) denial of use. It is important to note that a single event can result in the simultaneous occurrence of more than one of the outcomes. Figure 4 shows the outcome possibility matrix for the threat-asset combinations; a value of zero indicates that the outcome is impossible for that threat-asset combination, and a value of one means the outcome is possible for that threat-asset pair; greater granularity can be achieved by assigning values lying between zero and one, indicating varying degrees of possibility for the occurrence of each outcome.

The ideal safeguards system prevents the threats from attacking the assets and achieving the postulated outcomes. The safeguards system model consists of a set of safeguards functions for each of the distinguishable threat-asset pairs (nine T-A pairs, in this application) in such a way that the relative importance of each function within the set of functions for each T-A pair is about the same. Then, for each of the individual safeguards functions, a set of subfunctions provides performance criteria for the adequacy and completeness of that safeguards function; each of the subfunctions is devised so that the relative importance of each subfunction within a specific function is about the same. Again referring to Figs. 1-3, the figures show the safeguards functions and subfunctions for each distinguishable threat-asset pair.

	Unauthorized Access or Use	Modification or Tampering	Damage or Destruction	Disclosure	Theft	Denial of use
Natural Hazards – Facility	0	1	1	0	0	1
Natural Hazards – Hardware	0	1	1	0	0	1
Natural Hazards – Software	0	1	1	0	0	1
Natural Hazards – Documents/ Displays	0	1	1	0	0	1
Direct Human – Facility	1	1	1	1	1	1
Direct Human – Hardware	1	1	1	1	1	1
Direct Human – Software	1	1	1	1	1	1
Direct Human – Documents/ Displays	1	1	1	1	1	1

Fig. 4. Outcome Possibility Matrix for Computer/Information Security Application

LAVA evaluates the value of the assets to the organization in qualitative terms. The evaluation takes into account the criticality of the asset to organizational operations, the sensitivity of the asset to adversarial gain from theft or disclosure, and the necessity for the asset to maintain its integrity in terms of modification. The user may also specify monetary values for the asset to maintain its integrity in terms of modification. The user may also specify monetary values for the assets in any consistent currency system (LAVA's expertise does not extend to currency conversion).

Both government and corporate organizations may be the targets of a variety of hostile agents [35,36], and the intensity of the threat may change with time and circumstances. The dynamic threat strength can be analyzed if the subject system is extremely sensitive to a changing threat and if the subject organization has access to the kinds of information the analysis requires. The dynamic threat analysis takes into account possible threat agents and their potential attack goals with respect to the target(s) of the attack. The dynamic aspects of the natural hazards may or may not be of interest; these include both random natural hazards, such as volcanic eruptions or earthquakes, as well as the natural hazards more cyclic in nature, such as hurricanes, tornadoes, torrential rains, and the like. The human threat agents in each of the human threat categories all act for different reasons, so they may differ widely in motivation, capability, and opportunity. Similarly, the goals of the attacks may vary, but all categories of goals may be used by all categories of threat agents. Clearly, more than one of the goal categories may be the goal of a single attack, and a single attack may be perpetrated by more than one category of threat agent. Figure 5 illustrates the dynamic threat analysis structures. A more detailed discussion of the dynamic threat analysis can be found in References 25 and 26.

The impact analysis measures costs in both qualitative and quantitative terms: LAVA uses qualitative measures for intangible cost sources like loss of reputation or strategic posture, and quantitative measures for tangibles like repair/replacement costs or litigation costs.

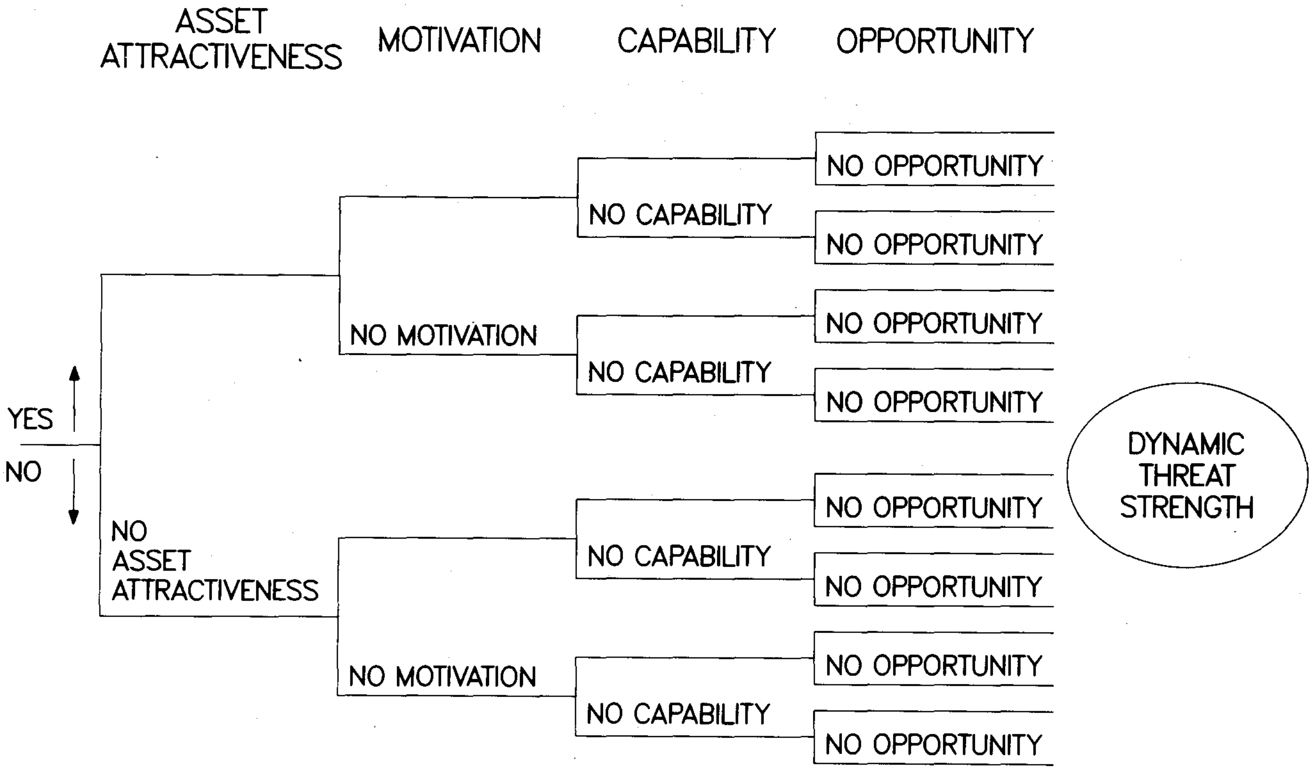


Fig. 5. Analysis Structure for Dynamic Threat

Loss exposure, or risk, results from a combination of asset value, threat strength, safeguards system weakness, and event costs. LAVA calculates both a monetary and a nonmonetary loss exposure measure for each [threat, asset, safeguards function, outcome, impact] combination. These loss exposure values can be aggregated in whatever ways are of interest to the user; less aggregation provides more information for specific decision making, but more aggregation provides a bottom line for upper management..

FEATURES OF LAVA/CIS VERSION 2.0

The long-awaited new version of the computer and risk assessment application of LAVA, LAVA/CIS Version 2.0, was released for the first time on a limited basis in April 1990. The new version has a much improved vulnerability assessment section, and has the additions of an asset-value estimation, a threat-strength estimation, and both monetary and nonmonetary (or intangible) impact analysis, expanding the LAVA 2.0 software engine into a full risk assessment package. This section discusses what the software is, what its operating requirements are, and how it is distributed.

What is LAVA 2.0 and what are its operating requirements? The LAVA 2.0 general software engine is a compiled, fully self-contained piece of software that runs on the IBM-PC class of personal computers. No additional software other than MS- or PC-DOS (version 2.0 or greater) is required to run LAVA 2.0. Minimum required hardware includes 1) 512 K available random-access memory, 2) a hard disk with about 1 megabyte of available space to store LAVA.EXE and the permanent application data sets, and 3) a floppy disk drive for the diskette holding the volatile application data sets. The report generators are compatible with a wide variety of dot-matrix, ink-jet, and laser printers.

What is new about LAVA/CIS 2.0? Instead of multiple code segments, LAVA 2.0 is integrated into a single menu-driven program; the menu items are selected with user-friendly light bars. Like previous versions, LAVA 2.0 applications are completely self-documented. In addition to the many definition and instruction screens, the LAVA 2.0 software engine now can display specific definitions selected as needed by the user during the progress of a LAVA assessment.

Besides an updated, much-improved vulnerability assessment (VA) portion, the new version includes a consequence analysis (CA) portion, making LAVA 2.0 a full risk assessment software system. The CA portion comprises an asset-value estimation, a threat-strength estimation, an outcome-severity mitigation estimation, and both monetary and nonmonetary (or intangible) impact analysis. The interactive vulnerability and consequence analysis questionnaire segments have hot keys for backing up in the questionnaire and for making a graceful emergency exit from the questionnaire if necessary. Both the VA and CA sections have independent report generators; the VA report format is fixed but has user-selectable graphic displays, and the entire CA format can be tailored by the user. The VA interactive, scoring, and reporting segments can be executed without doing the CA section. The interactive portion of the CA can be executed before, after, or at the same time as the VA; however, the CA scoring and reporting segments can not be run until after the VA has been completed.

In addition, a set of utility options permits the user to print unanswered questionnaires, partially answered questionnaires as memory refreshers in mid-assessment, fully-answered questionnaires at the completion of the VA for documentation purposes, and management worksheets for issue resolution. Finally, the LAVA 2.0 software engine now has color capabilities for those who have color monitors.

The data sets for LAVA/CIS Version 2.0 have been modified and expanded over those of Version 1.01. Some additional issues have been considered in the VA questionnaires, the security-requirement determination has been modified slightly, the underlying outcome set has been changed a little, and many of the VA questions have had their wording clarified. The definition screens have been reorganized so that there is only one definition per screen. All data sets for the CA portion are new.

All in all, the new LAVA 2.0 software engine is chock full of new features, all designed with the user in mind. Upgrading to the new computer- and information-security application, LAVA/CIS Version 2.0, should be very worthwhile!

How does one obtain LAVA/CIS? The Los Alamos National Laboratory is distributing the LAVA Software System for Computer and Information Security, LAVA/CIS Version 2.0, free of charge to Government agencies. It is available only to graduates of a LAVA training workshop–those who have faithfully attended and participated in the workshop. Because the workshops are an unfunded activity, there is a fee for the training workshops to recover workshop costs.

LAVA Workshops at Los Alamos and elsewhere. The LAVA/CIS Version 2.0 workshops, usually held at Los Alamos, last a full five days from 8:30 a.m. to 5:00 p.m. daily. The workshops present the LAVA philosophy and mathematical approach to vulnerability and risk assessment, and are hands-on workshops in which the participants complete a real assessment of a real computer installation. Attendance at all class sessions is required to graduate and receive the LAVA/CIS 2.0 software that is distributed to the graduates at the end of the workshop.

The workshops are intended for persons who have the responsibility for vulnerability and risk assessments in the computer- and information-security area; persons who require training in physical, technical, informational, and operations security activities; security auditors; and persons who manage security activities. The instruction staff provides help in how to use LAVA/CIS for vulnerability and risk assessments, as a training aid, as a preparation for security audits and compliance inspections, as a design tool, and as a decision aid.

If an agency wishes, the LAVA staff can hold a workshop/assessment at a site specified by the agency. The basic workshop content would be the same as those held at Los Alamos, but the agency could have as many participants as desired, and the workshop would produce a valid assessment of an installation belonging to the agency.

CONCLUSIONS

LAVA/CIS Version 2.0 is a comprehensive, rigorous, understandable approach to computer/information security risk assessment. It is a very affordable alternative to high-priced commercial risk assessment software. It can be used in-house by agency employees, obviating the need for the expensive services of outside consultants. Its flexibility in the order of execution of its various parts, in doing a stand-alone vulnerability assessment or a complete risk assessment, in doing either only nonmonetary impact analysis or both monetary and nonmonetary impact analysis, and in tailoring its reports contributes to its ease of use.

In addition, LAVA's capability to assess the dynamic aspects of the threat spectrum makes it an ideal tool for modelling applications of interest to the intelligence and military communities. It would also be highly applicable in the business community in situations ripe for industrial espionage.

Using the LAVA approach for risk assessment has benefits that do not accrue from the use of other methods. First, the automated report generators produce results that are immediately usable, both to managers who must make major, far-reaching decisions and to the security personnel in the field whose job it is to maintain an acceptable level of safeguards. Second, because LAVA produces both qualitative and quantitative results, users feel more comfortable with the results because they understand both the results and the information that produced those results. Third, because LAVA does not require the user to generate probabilities (often unfounded) for its operation but instead relies on a natural-language user-friendly interface to acquire its data, users are more willing to act upon its results. Fourth, LAVA includes a way to assess the changing, or dynamic, aspects of the threat spectrum. And finally, because of the team environment in which an assessment is performed and the discussions that arise among team members, using a LAVA application has proved to be an experience that both raises the security consciousness of the users and enhances the overall working environment at the facility.

REFERENCES

1. S. Katzke, "National Bureau of Standards Perspective on Risk Analysis: Past, Present, and Future," presented at the 1st Federal Risk Analysis Workshop, Montgomery, Alabama, January 1985.

2. S. T. Smith, "A Government-Wide Overview of Risk Analysis Methodologies," presented at the 8th DOE Computer Security Group Conference, Richland, Washington, April 1985.

3. S. T. Smith and J. J. Lim, "An Automated Procedure for Performing Computer Security Risk Analysis," in Proceedings 6th Annual ESARDA Symposium on Safeguards and Nuclear Material Management, May 1984, pp. 527-530.

4. N. J. McCormick, Reliability and Risk Analysis Methods and Nuclear Power Applications. New York: Academic Press, 1981.

5. W. D. Rowe, An Anatomy of Risk. New York: John Wiley & Sons, 1977.

6. M. D. Mesarovic, D. Macks, and Y. Takahara, Theory of Hierarchical Multilevel Systems. New York and London: Academic Press, 1970.

7. Y. M. I. Dirickx and L. P. Jennergren, Systems Analysis by Multilevel Methods. New York: John Wiley & Sons, 1979, pp. 10-82.

8. P. C. Fishburn, Decision and Value Theory. New York: John Wiley & Sons, 1964.

9. R. L. Keeney and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs. New York: John Wiley & Sons, 1976.

10. R. Schlaifer, Analysis of Decisions Under Uncertainty. Huntington, New York: Robert E. Krieger Publishing Company, 1978.

11. R. E. Bellman and L. A. Zadeh, "Decision-making in a Fuzzy Environment," Management Science, Vol. 17, No. 4, December 1970.

12. A. Kaufmann and M. M. Gupta, Introduction to Fuzzy Arithmetic: Theory and Applications. New York: Van Nostrand Reinhold Company, 1985.

13. L. A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility," Fuzzy Sets and Systems, Vol. 1, pp. 3-28, 1978.

14. C. V. Negoita, Expert Systems and Fuzzy Systems. Menlo Park, California: The Benjamin/ Cummings Publishing Company, Inc., 1985, pp. 52-58, 74-88, 95-112.

15. P. H. Winston, Artificial Intelligence. Reading, MA: Addison-Wesley, 1984, pp. 251-288.

16. R. Jain, "A Procedure for Multiple-Aspect Decision-Making Using Fuzzy Sets," Int. J. Systems Sci., Vol. 8, No. 1, pp. 1-7, January 1977.

17. P. J. H. Schoemaker and C. C. Waid, "An Experimental Comparison of Different Approaches to Determining Weights in Additive Utility Models," Management Science, Vol. 28, No. 2, February 1982.

18. E. M. Johnson and G. P. Huber, "The Technology of Utility Assessment," IEEE Trans. Sys., Man, Cyber., Vol. SMC-7, No. 5, May 1977.

19. L. A. Zadeh, K.-S. Fu, K. Tanaka, and M. Shimura (Eds.), Fuzzy Sets and their Applications to Cognitive and Decision Processes. New York: Academic Press, 1975.

20. S. Sudman and N. M. Bradburn, Asking Questions: A Practical Guide to Questionnaire Design. San Francisco: Jossey-Bass, Inc., 1982.

21. S. T. Smith and J. J. Lim, "An Automated Interactive Expert System for Evaluating the Effectiveness of Computer Security Measures, presentyed at the 7th Department of Defense/National Bureau of Standards Computer Security Conference, Gaithersburg, Maryland, September 24-27, 1984.

22. S. T. Smith, J. R. Phillips, R. M. Tisinger, J. J. Lim, D. C. Brown, and P. D. FitzGerald, "LAVA: A Conceptual Framework for Automated Risk Analysis," presented at the 1986 Annual Meeting of the Society for Risk Analysis, Boston, Massachusetts, November 9-12, 1986.

23. S. T. Smith, "LAVA: An Expert System Framework for Risk Analysis," presented at the 1st International Computer Security Risk Management Model Builders Workshop, Denver, Colorado, May 24-26, 1988.

24. S. T. Smith and J. J. Lim, "Framework for Generating Expert Systems to Perform Computer Security Risk Analysis," in Proceedings First Annual Armed Forces Communications and Electronics Association Symposium and Exposition on Physical and Electronics Security, August 1985, pp. 24-1–24-7.

25. S. T. Smith, J. R. Phillips, D. C. Brown, and P. D. FitzGerald, "Assessing the Threat Component for the LAVA Risk Management Methodology," in Proceedings 9th DOE Computer Security Group Conference, May 1986, pp.118-123.

26. S. T. Smith, "Risk Assessment and LAVA's Dynamic Threat Analysis," in Proceedings 12th National Computer Security Conference, October 1989, pp. 483-494.

27. S. T. Smith and J. J. Lim, "An Automated Method for Analyzing Computer Security Risk," presented at the 7th DOE Computer Security Group Conference, New Orleans, Louisiana, April 10-12, 1984.

28. S. T. Smith and J. J. Lim, "An Automated Method for Assessing the Effectiveness of Computer Security Safeguards," in Proceedings IFIPS Second International Congress on Computer Security, Toronto, Canada, September 1984.

29. S. T. Smith and J. J. Lim, "LAVA: An Automated Computer Security Vulnerability Assessment Software System (Version 0.9)," Los Alamos National Laboratory document LA-UR-85-4014, December 1985.

30. S. T. Smith et al., "LAVA for Computer Security: An Application of the Los Alamos Vulnerability Assessment Methodology," Los Alamos National Laboratory document LA-UR-86-2942, 1986.

31. S. T. Smith and J. J. Lim, "Assessment of Computer Security Effectiveness for Safe Plant Operation," Trans. Am. Nucl. Soc., Vol. 46, pp. 525-526, 1984.

32. S. T. Smith, J. J. Lim, and J. Lobel, "Application of Risk Assessment Methodology to Transborder Data Flow,"in Handbook on the International Information Economy, Transnational Data Report, Springfield, VA (November 1985).

33. S. T. Smith and R. M. Tisinger, "Modeling Risk Assessment for Nuclear Processing Plants with LAVA," Nucl. Mater. Manage., Vol. XVII, pp. 101-104, 1988.

34. S. T. Smith et al., "LAVA for Computer Security: An Application of the Los Alamos Vulnerability Assessment Methodology, Release Version 1.01," Los Alamos National Laboratory document LA-UR-86-2942,September 1987.

35. N. R. Bottom, Jr., and R. R. J. Gallati, Industrial Espionage: Intelligence Techniques and Countermeasures. Boston: Butterworth Publishers, 1984.

36. R. Eells and P. Nehemkis, Corporate Intelligence and Espionage: A Blueprint for Executive Decision Making. New York: Macmillan, 1984.

WORKFLOW: A METHODOLOGY FOR PERFORMING A QUALITATIVE RISK ASSESSMENT

Paul D. Garnett
SYSCON Corporation
Route 206
Dahlgren, Virginia 22448

INTRODUCTION

The methodology described in this paper for performing a qualitative risk assessment was developed for an organization involved in the generation and maintenance of computer software and firmware. The environment in which software and firmware is designed, developed, and maintained has a direct bearing on the quality, trustworthiness, integrity, and freedom from malicious code of the end product embedded system.

This paper will describe a methodology which enables the organization to look at how work flows in the organization, hence the name "Workflow," and to look for weaknesses in the system which would allow a sufficiently motivated saboteur to exploit these vulnerabilities. In many cases, organizational vulnerabilities occur where adequate checks and balances do not exist. By analyzing how work flows within an organization, the lack of these checks and balances becomes obvious to the analyst performing the workflow.

For any company that uses computers, in particular, those companies who produce and/or maintain software and firmware, the environment in which this work is performed is constantly changing. The field of software/firmware development and the use of computers in the workplace becomes more dynamic every year. We need to look at our existing controls and ask ourselves, "If one person goes bad, will our entire system collapse?" Also, if it were known that someone in the organization had a hostile intent, is there anything that would be done differently?

THE WORKFLOW PROCESS

PRELIMINARY EFFORTS

Performing a workflow analysis begins with a briefing to upper management to let them know what is going to be done, how long it will take, and the degree of involvement of all levels of personnel within the organization to be analyzed. It is imperative that upper management endorse and support the project. One reason for this endorsement is that performance of a workflow analysis requires the cooperation of certain individuals within the organization.

The actual performance of the workflow starts with a review of a standard list of discussion items, summarized in Figure 3. These discussion items should be reviewed to ensure that recent developments in the computer field are included. Since this technology is advancing so quickly, the list of discussion items used for a workflow analysis performed one month might need to be updated for an analysis to be performed a few months later.

Once the discussion items have been finalized, copies should be prepared so that each interviewee can have his own list of the items to be discussed. This list is NOT to be used as a questionnaire; rather, the list of discussion items is a guide to be used during a free-form discussion between two people: the interviewer and interviewee.

Before any of these interviews begin, a careful determination must be made by the people performing the workflow analysis and by upper management of the organization to be analyzed on who ought to be interviewed. Usually, these people include the first and mid-level managers involved in the day-to-day operations of the company. These are the people who understand how work is REALLY performed, rather than what upper management expects or what is represented by the official organization charts.

In addition to the first and mid-level managers, interviews should be held with specific technical experts and those people holding critical positions. Examples of these positions include managers of computer systems; security personnel including physical and computer security; and people in the personnel department who are familiar with various organizational procedures.

THE INTERVIEWS

The individual discussions have proven to be invaluable in soliciting information which might otherwise remain buried within an organization. First, many people are reluctant to share sensitive information when filling out a questionnaire which will be read by their superiors in the organization. This includes information which shows how the individual or the organization may have done something wrong, stupid, or otherwise contrary to standard computer security practices. Also, people may not divulge information when being interviewed by a panel of people, or even by two people working together. However, in a private setting, once a rapport has been established between the interviewer and interviewee, it is amazing what some of the people will talk about, especially near the end of the discussion when the interviewee is asked if there are any specific areas of concern about how things are done or not done in the company.

A typical discussion, one in which the majority of discussion items are touched on, may last from one and a half to three hours. During this time, the interviewer will be taking copious notes while steering the conversation into subject areas that may lead towards a discussion of potential security problems. It does not matter whether or not the discussion items are discussed exactly in order as specified in Figure 3. What is important is that the conversation flows and that all the discussion items are covered at some point during the interview. It is up to the interviewer to ensure that each item is discussed, even if it is only to ask whether a particular topic applies to the person being interviewed.

INDIVIDUAL REPORTS

Once an interview is concluded, the interviewer/analyst must transcribe the notes taken as soon as possible. This is best done when the material is still fresh, so that the context of what was being talked about can be easily recalled as the information is being analyzed. For each person interviewed, an individual report is generated. This report is based solely on the information gathered during that one interview. Part of the value in performing a risk assessment using the workflow methodology is that these individual reports, once completed, can be taken back to the interviewee and the information validated by the person who gave the information. So often during an interview, words, concepts, and ideas get slightly scrambled during translation into a formal report and it is imperative that the person who gave this information have an opportunity to correct any misinterpretations, assumptions, or anything at all which is not perfectly correct in the individual report.

The concept of this individual report and the opportunity for the interviewee to review it before any of the information is passed on to other people is also invaluable in persuading someone to give information which they feel uncomfortable about discussing. This methodology allows the individual to review what they have said and indicate, if desired, that certain information is NOT to go forward in any further reports. In some cases, people have reviewed their individual report and determined that certain sensitive information be slightly modified to soften the effect. In other cases, it has been requested that certain information be treated as privileged and that it not be included in the overall report, a compilation and interpretation of all the individual reports.

These individual reports are generated as follows. First, the rough notes taken during the interview must be deciphered. During the course of an interview, 10 to 15 pages (8.5 x 11) of notes might be taken. The sheer volume of material, written in haste during the interview, causes many of the inaccuracies uncovered during the review to be introduced. The first job of the analyst is to take each piece of information, line by line through the notes, and put it into one or more of the categories listed in Figure 1. This is best done using some sort of automated database system. There are many databases designed to run on personal computers which can be used in organizing this material. It would help, however, to use a database system which allows free form information to be entered and does not require prespecified field lengths.

1. ORGANIZATION NAME:
2. HEAD OF THE ORGANIZATION:
3. ORGANIZATION CODE:
4. REPORTS TO:
5. DISCUSSION DATES:
6. STAFF SIZE:
7. NUMBER OF ORGANIZATIONAL SUBUNITS:
8. STAFF:
9. SECRETARIAL STAFF:
10. STAFF RESPONSIBILITIES:
11. SUBUNIT #1 - NAME:
    LEADER:
    SIZE:
    MISSION:
    TASKS:

    SUBUNIT #2 - NAME:
    LEADER:
    SIZE:
    MISSION:
    TASKS:

    . .
    . .
    . .

    SUBUNIT #N NAME:
    LEADER:
    SIZE:
    MISSION:
    TASKS:

12. KEY PEOPLE:
13. ORGANIZATION MISSION:
14. ORGANIZATION INPUT:
15. ORGANIZATION PRODUCTS:
16. ADDITIONAL ORGANIZATION TASKS:
17. REFERENCE DOCUMENTS:
18. TRAINING:
19. ROTATIONS:
20. COMPUTER PROGRAMS:
21. COMPUTER USAGE:
22. EQUIPMENT CONFIGURATION CONTROLS:
23. CONTROLLED ACCESS:
24. CONTRACTORS:
25. INTERFACES:
26. OUTSIDE PARTICIPATION:
27. METHODOLOGIES:
28. REVIEWS:
29. ARCHIVES:
30. BACKUPS:
31. AUDITS:
32. CLASSIFICATION:
33. SENSITIVE AREAS:
34. KNOWLEDGE LIMITATIONS:
35. FIRE DRILLS:
36. ACTION:
37. VULNERABILITIES:
38. RECOMMENDATIONS:
39. COMMENTS:

Figure 1. INDIVIDUAL REPORT CATEGORIES

Each individual report is simply a printout of the information gathered during the interview which the analyst has entered into the database. The review of this report by the interviewee is best done in the presence of the interviewer because misconceptions, errors, and so forth can be discussed and corrected immediately by marking up a copy of this report. Also, as discussed earlier, any pieces of information which are not to be included in the final report based on the interviewee's request can be bracketed and marked with the statement "PRIVILEGED INFORMATION." A request by an interviewee to withhold any information must be honored because this trust, once broken, cannot be regained. Also, if this trust is broken, it is not likely that anyone else in the organization will be willing to open up and have good, honest discussions upon which the entire workflow analysis depends.

THE FINAL REPORT

Once all of the individual discussions have been held and the reports have been generated and verified, it is time to collate all of the information which has been gathered. The first step in performing this aspect of the workflow analysis is to take, from each individual report, information from each category and generate category subdocuments. These subdocuments will include everything that was said by all participants on any one topic.

From the category subdocuments, the analyst must decide, for each paragraph, where this information is to be placed in terms of the chapters of the final report. A sample listing of chapters is presented as Figure 2. Each application of the workflow methodology will require a final report format tailored to the specific organization being analyzed.

1. ANALYSIS TOOLS
2. AUDITS
3. BACKUPS
4. CLASSIFICATION
5. CONFIGURATION MANAGEMENT/ QUALITY ASSURANCE
6. CONTINGENCY PLANS
7. CONTRACTORS
8. DOCUMENTATION
9. EMBEDDED SYSTEM SOFTWARE OPERATIONS
10. EXAMPLES OF COMPUTER MISCHIEF/ SABOTAGE/CRIME
11. FIELD OPERATIONS
12. FIRE DRILLS
13. FOREIGN AND DOMESTIC THREATS
14. GENERAL PURPOSE COMPUTER OPERATIONS
15. HARDWARE
16. ILLICIT CODE
17. KEY POSITIONS
18. MEDIA PREPARATION FACILITIES
19. NETWORKS
20. PASSWORDS
21. PERSONAL COMPUTERS
22. PERSONNEL
23. PHYSICAL SECURITY
24. RISK ASSESSMENT
25. TRAINING

Figure 2. FINAL REPORT - CHAPTER TITLES

At this point one might ask, why go through all of the trouble preparing individual reports based on the categories as defined in Figure 1 when all of this information will be reorganized later into the chapters of a final report. Also, why not design the list of discussion items to make the final report generation easier by asking questions directly related to chapter titles. Reasons for this approach are as follows. The list of discussion items is designed to elicit information and to establish a rapport between the interviewer and interviewee, NOT to make the analyst's job easier. The categories of information used in the individual reports are designed to take the analyst one step away from the discussion items towards an organization of information related to the format of a final report, but still close enough to the original discussion items to enable the interviewee to understand where the information came from in terms of their discussion. Finally, part of the value in performing the workflow and in analyzing the information collected lies in the processing of this information. The more times the analyst looks at the information and works with it in different ways, the more likely that insights will be gained on how the organization functions and how the key individuals perceive the organization as functioning. It is in these insights by the analyst that the value of performing the workflow is found.

A general example of this depth of analysis is when several people address similar situations in an organization. This information will get put into the same category in each individual report. Once this information is collected in the category subdocuments, it becomes obvious that different people are seeing the same situation in different ways; in some cases, this can show where the right and left hands of an organization do not really know what the other is doing.

Once paragraphs have been assigned to specific chapters of the final report, an outline is written. This is done by ordering these paragraphs within each chapter. Once this is done, each chapter is written. Since the paragraphs within a chapter came from different reports and are not always complete thoughts and ideas, they need to be molded into cohesive language.

For each chapter, primary recommendations, secondary recommendations, and summary information are chosen from the chapter material. Chapter summaries will include the most important information that the analyst wants seen by the reader. Primary recommendations are those which the analyst feels very strongly about and is willing to say unequivicably that they should be accomplished to ensure the security posture of the organization. Secondary recommendations are those worthy of very serious consideration. These recommendations, along with the chapter summary, are included at the end of each chapter of the final report.

During the writing of the individual reports and the generation of the final report, the analyst should be jotting down notes which will later be included in the final document introduction. This introduction will include general comments about the conduct of the analysis, any caveats about the meaning of certain information and results, and any other thoughts and insights which the analyst came up with during the analysis itself and the preparation of the documents.

To create the executive summary, information in each chapter's summary which the analyst feels is most important is extracted as well as the chapter's primary recommendations. This information is then outlined and re-written as required.

Since the final report will undoubtedly be a rather thick document, it is very worthwhile to present the primary findings of the study as a briefing to the management personnel who originally requested the workflow analysis. These viewgraphs are essentially complete upon the completion of the final report, as a high level briefing can be put together very easily from the executive summary, or a more detailed presentation can be generated from the material in each chapter's summary and recommendations sections.

DISCUSSION ITEMS

Although it is unimportant in which order the discussion items are covered, it IS important to discuss paragraph 1 ("Why are we having this discussion?") first. This gives the interviewer a chance to state the ground rules of the discussion, and gives the interviewee a chance to get comfortable with discussing some possibly sensitive topics.

In addition to the obvious value to the interviewer in holding these discussions to collect information as part of the workflow analysis, these discussions are valuable to the organization being studied because the people being interviewed will gain a new appreciation of computer security issues and a greater awareness of actions which they themselves can take to improve the security posture of their organization. In some cases, these discussions are the first time that people have ever had a good understanding of what malicious code is and what the differences are between viruses, worms, trojan horses, trapdoors, and so on.

The explanations of the discussion items in Figure 3 are not intended in this paper to be a complete expansion of the specific item being discussed. These are only sample questions, not all inclusive by any means. They are only being provided here to give the reader an idea of the types of directions that a discussion can proceed under each of the topics.

Some of these discussion items may not appear to have anything to do with computer security but are included to give the analyst more of an understanding of what the organization does; how they do it; who they interface with, when, and why; and, in a nutshell, the required understanding of what makes the organization tick. The analyst needs to understand the philosophy of the organization and what the social climate is in the organization being studied. It is from these understandings that analytical insights come and significant, RELEVANT recommendations can be made to improve the organization's security posture.

1. Why are we having this discussion?

    a. History/Background - The interviewer presents basic information on who asked for the study, how it is being done, why it is being done, and what is hoped to be accomplished by performing the study.

Figure 3. DISCUSSION ITEMS

b. Collecting Data - The interviewer discusses the format of the study, including collecting data during these individual discussions, preparing the individual reports, validating the information, and collating all the individual reports into a final report. This is also the point at which the structure of the interview itself is discussed, and the statement is made that the discussion need not follow the list of discussion items, that the topics can be discussed in any order, and that the list is only a checklist to make sure that all the topics are covered sometime during the discussion.

c. Recommendations - The purpose of performing the workflow analysis is to come up with recommendations on how to improve the organization's security posture.

d. On/Off Record - The interviewer states very clearly that if the discussion goes into subject areas which the interviewee is uncomfortable about having appear in a final report, that he/she needs only to make this clear and the information will be treated as privileged between the interviewer and interviewee.

2. What work is performed in your organization?

   a. Organization Chart - Collect and discuss any organization charts which the interviewee may have.

   b. How? - How is work performed in your organization? What are the people really doing?

   c. By whom? - Who are the people in the organization? Which people perform which tasks?

   d. How learned? - How are people taught their tasks? Is it all on-the-job training? Are there documents which describe the work to be performed? Is learning all word of mouth? And so on.

3. Workflow

   a. Organizational Inputs - What are the inputs into the organization that drive the work being performed?

   b. Organizational Products - What do you produce? This could be specific software, services, documentation, and so forth.

4. What major computer programs are:

   a. Used? - What computer programs do you use to perform your tasks?

   b. Developed? - Does your organization develop any software or firmware? If so, what are they, how do they work, what are they used for, etc.?

   c. Maintained? - Does your organization maintain any software or firmware? If so, how is this done, according to what procedures, etc.?

5. What computer networks are:

   a. Used? - Do you use any computer networks in the course of doing business?

   b. Developed? - Discuss the details of the networks, the methodologies used in designing and developing them, etc.

   c. Maintained? - Discuss the details of the networks, the procedures followed, if any, in maintaining them, etc.

Figure 3. DISCUSSION ITEMS (Continued)

6. Any illicit code in the organization?

   a. Viruses - Do you know of any incidents in which viruses have been found in the organization? It may be necessary here to discuss the differences, technically, between viruses, worms, trojan horses, time and logic bombs, etc.

   b. Trojan Horses - Do you know of any incidents in which a trojan horse was introduced into any software or system used, developed, or maintained by the organization?

   c. Other (trapdoors, logic bombs, time bombs, worms, etc.) - Do you know of any incidents in which any of these examples of illicit code were used in your organization?

7. What organizational controls are there?

   a. Reviews - For organizations which develop software, firmware, networks, systems, etc., are there design reviews held? How are they structured? How often are they held? When during the life cycle of the system are they held? For production work in using computer systems, is there any kind of review process to see how employees actually use the system(s)?

   b. User/Programmer/System Operator Restrictions - Are there any restrictions on how these categories of system users are restricted in their use of any system to which they have access? If so, what are they, how do they work, and so on.

   c. Source/Object/Documentation - What are the controls placed by the organization on source code? Who can change it? How? When? Similarly, what are the controls on object code and documentation? Where are they stored? Are there mechanisms in place to determine if any unauthorized modifications have been made?

   d. System Configuration - Are there any controls to prevent or detect unauthorized changes to computer system configurations? "Configuration" can mean connectivity to other system components, or simply how the system itself is set up in terms of default values.

   e. Backups/Audits - What are the details of all backup procedures on all systems for which your organization is in control or involved? Also, what are the procedures used, if any, in auditing system usage for all systems for which your organization is in control or is involved? Even if the organization is a user of a system and is not in control of the audit and backup processes, it is valuable to learn the perception of the user as to what he/she THINKS is being done by the owner of the system in terms of backups and audits. It is here that the analyst often finds out that the right hand of the organization does not know what the left hand is doing. In some cases, system users have stated that all backups are done by system personnel, while the system owner has stated that all backups are the responsibility of system users. This kind of an organizational disconnect is not as rare is it should be and can obviously lead to a disaster.

8. What is classified/unclassified?
   What is sensitive/company proprietary/not sensitive?

   a. To what level? - First, does the organization handle any sensitive or classified information at all? If so, to what level? What types of information are classified or sensitive? How is this material controlled? Is the material in hardcopy form? Is it used in computer systems? Which ones?

   b. Programs/Data/Documents? - If company proprietary, sensitive, or classified information is handled by the organization, what form does it take and what controls are in place to deal with it?

Figure 3. DISCUSSION ITEMS (Continued)

c. Access to Vaults/Laboratories/Terminal Rooms - Does the organization have any "special" areas? What are they? Which people are allowed in these areas? How is access controlled?

9. Use of contractors/subcontractors

a. Which ones? - Which other organizations do you deal with? What are the arrangements? How well do you get along? What sort of relationships have been established?

b. What products? - What do you get from these other organizations? In what form? What checks are performed on deliverables received?

c. Who interfaces? - Who are the people in your organization who actually deal with these external organizations?

d. Consultants? - Do you ever use consultants? What controls do you place on them, if any? What do you use them for? What do they know about the details of your organization?

10. Contacts with (other) government agencies

a. Which ones? - Which government agencies does your organization deal with, if any?

b. Who interfaces? - Who are the people in your organization who have contact with these agencies?

c. When? - Is there a regular cycle to these meetings? Is there a specific point during any work cycle that these organizations are contacted?

d. Why? - What is the purpose of the contact with these specific agencies?

e. What products are received/delivered? - Are there any products received or delivered to these agencies? If so, what are the details of these interactions?

11. What are the sensitive or critical areas of knowledge?

a. Are there controls to limit one person's grasp of the full picture? - In many organizations, the more knowledgeable employees become, the more valuable they are to that organization. However, as employees become more knowledgeable about company operations, they become more of a potential threat to the organization if that knowledge were to be used against the organization. For example, someone intent upon introducing malicious code into a system may take a job with the group that performs independent testing on the system as it is being developed. If this individual then transfers into the group that writes the code, he or she may know how to put something into the system which will never be discovered because he/she knows there are no tests to find it.

b. Are assignments varied/rotated? - Does the organization rotate personnel through various groups as part of a training program? Are people allowed to transfer between organizational subelements? Are people stymied when they want to move on to something more challenging?

c. Personnel Backups - For every critical job, is there more than one person fully capable of performing all the tasks?

d. What efforts are subject to "Fire drills?" - When do employees go into "panic mode?" Which positions are subject to high pressure and rush jobs? In many organizations, when push comes to shove and something

Figure 3. DISCUSSION ITEMS (Continued)

MUST go out the door, controls and procedures are bypassed in the interest of getting the job done. When this occurs, the organization becomes vulnerable to anyone willing to take advantage of the reduced controls and the willingness of people under pressure to overlook a system anomaly which ordinarily would be questioned.

12. Other

   a. Areas of discussion - Are there topics that have not been discussed that the interviewee feels ought to be talked about?

   b. Other people to talk with - Are there other people in the organization who the interviewee thinks would be valuable for the interviewer to talk with to learn more about specific areas which were discussed during the interview?

   c. Specific areas of concern - Are there any specific concerns which the interviewee has on any topic which might possibly relate to the security posture of the organization? Are there concerns which interviewees have that they have never felt comfortable in passing up the line which they would like to communicate to upper management anonymously? This is where the trust between the interviewer and interviewee comes into play. If there IS information to be passed up the line anonymously, and in many cases this has occurred, it MUST be communicated in such a way in the final report that the comment cannot be traced back to the individual who made it. This last discussion item often gives great insights into the real mood of the organization.

Figure 3. DISCUSSION ITEMS (Continued)

KEY POINTS IN PERFORMING A WORKFLOW

There are several hurdles to overcome in successfully performing a workflow analysis. First, trust must be established between the interviewer and interviewees. The anonymity of the individual discussions MUST be sacrosanct.

There must be a non-adversarial relationship established between the interviewer and interviewees. By this it is meant that the interviews must be held in a non-threatening manner. When someone states that they never can find the time to perform system backups, that they suggest to their users that they use their first names as passwords, and that there are multiple dial-in ports into their system, one does not immediately scream "You idiot, do you realize what you are letting yourself in for?" The analyst must take notes on the information and lead the discussion further into these problem areas to determine, if possible, WHY these policies are being followed so that later, in the individual and final reports, recommendations can be made to improve these potentially serious situations.

Another problem often encountered in performing a workflow analysis is the "It can't happen here" syndrome. Again, rather than getting into a heated discussion about "Oh yes it can, you've just been lucky so far," the analyst must work around this mindset and extract as much information as possible from this worry-free individual. What can be done in the final report is to make a general statement that some employees in the organization seem to hold the belief that they are immune to problems which have in some cases brought havoc to other organizations and that maybe some form of training may be in order.

One way to gain the cooperation of participants in this study is to let the participants themselves take the credit for any improvements in the organization's security posture. This can be done by keeping the results of the study low-key and relatively unpublicized. It is amazing to watch an organization begin to quietly implement some of the study's recommendations while publicly stating that there was nothing in the report which would indicate that they have any problems.

SUMMARY

Performing a workflow gives an organization insights into where they may be vulnerable to computer security weaknesses. A workflow analysis begins with a review of the discussion items to ensure that it is up-to-date in relation to the state-of-the-art in computer security threats and safeguards.

Once the discussion items have been finalized, briefings are provided to upper and mid-level management to inform them how the workflow project is to proceed and what cooperation will be required from whom.

Individual discussions are held with key people in the organization; individual reports are generated from these discussions according to a pre-organized report format/database; and these reports are reviewed with the individuals for accuracy. Once all of the key people have been interviewed and their individual reports verified, the information in these reports is collated according to general topics and a final report is generated.

The final report consists of discussions on many computer security related and organizational dynamic topics, and each chapter contains primary and secondary recommendations, and a summary. The executive summary of the final report consists of the key items in each chapter's summary and primary recommendations.

Briefings on the workflow analysis results can be generated easily based upon the summaries and recommendations listed in the report.

The great value in producing a workflow report is that the organization gains significant insights into its operations, the people within the company who participated in the workflow gain computer security awareness, and previously ignored or unrecognized vulnerabilities can be addressed. Since the workflow analysis draws its input from the people actually involved in the company's operations, the recommendations made in the study's final report are extremely relevant to the organization which was studied.

CRITICAL RISK CERTIFICATION METHODOLOGY

NANDER BROWN

U. S. SMALL BUSINESS ADMINISTRATION
OFFICE OF INFORMATION RESOURCES MANAGEMENT
1441 L STREET, N.W.
WASHINGTON, DC 20416

ABSTRACT

This paper describes an approach for evaluating, certifying, and accrediting very large automated systems. This approach can be used by Federal agencies to comply with the certification requirements of OMB Circular A-130. The Critical Risk Certification Methodology (CRCM), provides a cost effective and structured approach that agencies can use to evaluate and certify large aggregated systems. Such systems may have been previously ignored due to their size, complexity, and the large amount of resources required to evaluate and certify them.

The CRCM focuses on "critical risks" at various hierarchical levels within an application. Software tools are used to assist in the technical analysis, and to provide key information that can be used to support the decision to certify or not to certify the system. The CRCM review provides evidence that control and security measures have been implemented to ensure that information processed will be accurate, reliable, and the system will be available, even in emergency situations. The certifying official will have sufficient system security evidence to make a supportable recommendation to the designated system accrediting official.

Because of the structured process of CRCM, Agency certification and accreditation officials should be willing to accept more risks. This acceptance is based on the analytical evidence that most major security and control weaknesses have been identified, and appropriate corrective measures have been implemented, or are in the process of being implemented.

An example of how this methodology was implemented at the Small Business Administration (SBA) is used to describe the overall process.

INTRODUCTION

OBJECTIVE

To provide a cost effective evaluation process for the certification and accreditation of SBA's aggregated sensitive information systems; and to comply with OMB regulations A-130 and A-123 which directs Federal Agencies to conduct security reviews and internal control reviews of automated applications.

PROBLEM

SBA currently operates 15 sensitive information systems. Several large systems are aggregates of other systems. For example, the Loan Accounting System (LAS) is an aggregate system which includes three major applications: (1) Loan Origination and Disbursement, (2) Loan Accounting and Collection Processing, and (3) Loan Servicing and Debt Collection. These applications include approximately 24 individual application subsystems.

Traditional security certification review procedures normally require an evaluation of each individual subsystem. This review process is time consuming and very costly. The review is normally conducted by two qualified security personnel, and requires 4 to 6 weeks to complete per application subsystem. Additional time is required to prepare written reports and communicate findings and recommendations to appropriate agency officials. Many of these findings and recommendations may be trivial or low risk, but never the less, they must be documented and discussed. Finally, additional time is required by agency officials to discuss, prioritize, and prepare an action plan for correction of security weaknesses.

The SBA calculated the cost of reviewing each subsystem at an average cost of $15,000 to $20,000. The approximate cost to review and certify LAS would be $400,000. It would also require at least 3 fiscal years, based on managements budget priority for computer security. In order to find a more cost effective and timely way to certify LAS, and comply with Federal and SBA regulations, we needed a more efficient, but effective, way to achieve this goal. The CRCM was developed as a solution to this problem. In the process of formulating this approach, we researched many technical references and reviewed certification approaches used by other agencies.

Figure 1 provides an overview of the CRCM. Figure 2 provides an illustration of an aggregate application structure cross referenced to the CRCM review components.

CRCM PROCEDURE

The Critical Risk Certification Methodology includes tailored reviews of computer security of aggregated sensitive applications at various hierarchical levels. The review process focuses on "critical risks" at each level. A specific review procedure is developed to achieve a designated security and control evaluation. Each lower level is accorded a more detailed evaluation than its parent level. The CRCM review procedures include the following:

1. High level risk assessment (HLRA) of the computer security workload. The HLRA determines workload units and degree of risk associated with each activity. Selected sensitive systems are evaluated in more detail for certification and accreditation. The results of this review is an inventory of the organization's computer security workload in risk priority order.

2. System level risk assessment (SLRA) of the aggregate system. The SLRA includes detailed assessment of designated pervasive risks and special risks at the major system level.

   Five pervasive risks and three special risks are assessed. The results of this review is very similar to the results from step 1 above except that risk ratings are obtained for each subsystem within each major system of the aggregate system.

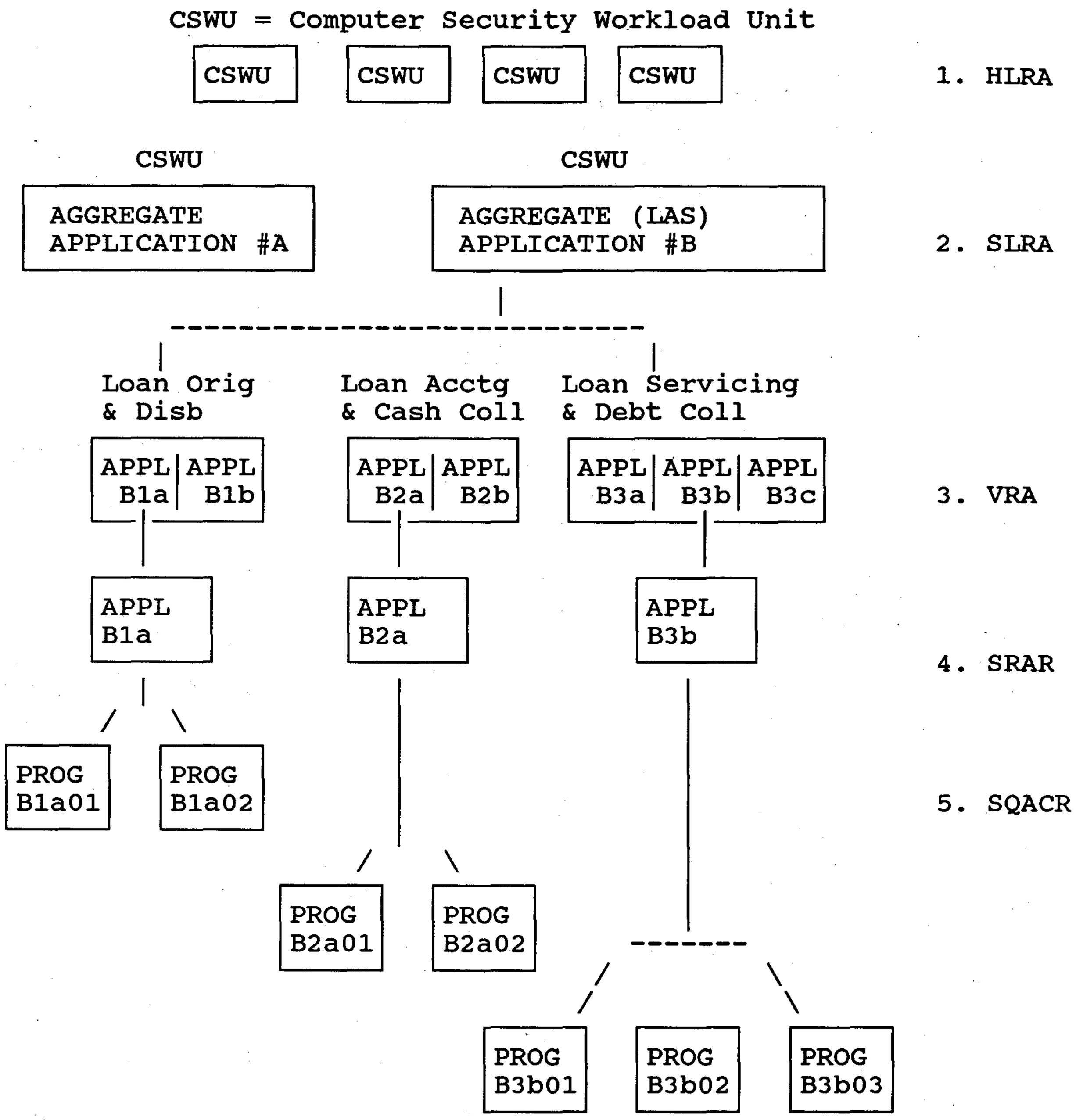
3. Vulnerability risk assessment (VRA) of the major systems. This assessment includes the identification of vulnerabilities and implemented control measures for each major system. An application control matrix is prepared for each major system. Detailed steps and tools include the use of software for the assessment of risk areas, and the review of control objectives based on the Model Framework for Management Control Over Automated Information Systems, published by the President's Council on Integrity and Efficiency (PCIE).

4. Security requirements analysis review (SRAR) of the highest risk application from each subsystem in step 3 above. This review is a traditional security review frequently performed by computer security personnel. The controls evaluated are determined by the sensitivity level of the aggregate system. A Security and Integrity Requirements Worksheet is prepared. It is the baseline that is used to evaluate installed security and controls. The report from this review provides detailed discussion of findings and recommendations.

5. Security and quality assurance compliance review (SQACR) of the most critical program modules from the high risk applications identified in step 3. This review includes the evaluation of functional system requirements, application design standards, quality assurance standards, and security and control requirements at the program module level. A detailed Application Review Questionnaire is completed based on interviews and analysis of program source code.

```
                ---------------------------------
               | HIGH LEVEL RISK ASSESSMENT of   |
   --------    | computer security workload for  |    Prioritized list of
  | CRCM-1 |-> | the organization. Work units    | -> security work units
   --------    | are ranked according to         |    according to risks
   (HLRA)      | pervasive risk areas and        |    ------------------
               | specific risk areas.            |
                ---------------------------------

                ---------------------------------
               | SYSTEM LEVEL RISK ASSESSMENT of |
   --------    | risks in the major system, and  |    Prioritized list
  | CRCM-2 |-> | subordinate subsystems. Each is | -> of application sub-
   --------    | assessed according to pervasive |    systems according
   (SLRA)      | risk areas and specific risk    |    to risks
               | areas.                          |    ------------------
                ---------------------------------

                ---------------------------------
   --------    | VULNERABILITY RISK ASSESSMENT   |    Matrix of security
  | CRCM-3 |-> | by matrix analysis of control   | -> techniques
   --------    | objectives, security techniques |    implemented by
   (VRA)       | and vulnerabilities             |    control objectives
                ---------------------------------     ------------------

                ---------------------------------
               | SECURITY REQUIREMENTS ANALYSIS  |    Documented list of
   --------    | REVIEW of high risk subsystems. |    controls based on
  | CRCM-4 |-> | Security and control techniques | -> system sensitivity
   --------    | reviewed for effectiveness and  |    and security
   (SRAR)      | appropriateness to the risks    |    requirements
               | identified in CRCM-1 and CRCM-2 |    ------------------
                ---------------------------------

                ---------------------------------
               | SECURITY AND QUALITY ASSURANCE  |    Specific findings
   --------    | COMPLIANCE REVIEW of critical   |    on compliance
  | CRCM-5 |-> | software. The review focuses on | -> with standards and
   --------    | compliance with security and    |    QA requirements.
   (SQACR)     | quality assurance standards.    |    ------------------
                ---------------------------------
```

Figure 1. CRCM Overview



Critical Risk Review Components:

1. HLRA - Computer security annual workload for the Agency
2. SLRA - Aggregate Application #B
3. VRA - Major systems B1, B2, B3
4. SRAR - Subsystems B1a, B2a, B3b
5. SQACR - Program modules:
   - B1a01 (Application B1a, Major System B1)
   - B2a02 (Application B2a, Major System B2)
   - B3b03 (Application B3b, Major System B3)

Figure 2. CRCM aggregate application structure

PARTICIPATION

The review process requires participation by the following groups:

- An independent review team
- The Agency Computer Security Program Manager
- User functional program managers
- Application system managers
- System design team leaders
- Selected computer programmers
- Computer operations personnel

CRCM REPORTS

Reports are prepared for each level of review. The reports provide specific findings related to the type of review conducted. The overall final report includes:

- An executive summary,
- Findings and recommendations codified by risk level
- A CRCM risk summary, and
- Certification recommendations.

The CRCM risk summary highlights each major risk/threat situation, control objective weaknesses, and the potential adverse impact of the control weakness on the organization. This summary is illustrated on the next page.

The Certification report discusses two levels of certification recommendations:

- Level 1 addresses control enhancements required to correct significant risk areas related to systems integrity, information disclosure, and degradation of service.

- Level 2 addresses control enhancements required for the system to meet generally accepted systems integrity standards.

Level 1 control recommendations should be implemented before the system is certified. These controls may require re-evaluation before the official certification and accreditation can be completed.

Level 2 control may be scheduled as part of a system enhancement project which will be implemented at a later date. Certification documents should note that additional security enhancements are necessary, but not serious enough to withhold certification and accreditation.

CRCM RISK SUMMARY REPORT

RISK/THREAT DESCRIPTION	CONTROL OBJECTIVE WEAKNESS	PROBABILITY*	EXPOSURE**	PRIORITY VALUE	BUSINESS IMPACT

Describe potential adverse impact to the organization in terms of the consequences of the risk/threat realization. Discuss the impact in dollar losses, information disclosure, information integrity, and/or system denial. Losses may be estimated and expressed quantitatively using the following order of magnitude scale:

0 =	Negligible (about $1)
1 =	on the order of $10
2 =	on the order of $100
3 =	on the order of $1,000
4 =	on the order of $10,000
5 =	on the order of $100,000
6 =	on the order of $1,000,000
7 =	on the order of $10,000,000

PRIORITY VALUE DETERMINATION

PROBABILITY	EXPOSURE		
	Low	Medium	High
Rare	1	3	6
Moderate	2	5	8
Frequent	4	7	9

Level 1 risks = priority values 7 to 9
Level 2 risks = priority values 1 to 6.

* Rare, Moderate, or Frequent
** Low, Medium, or High

SMALL BUSINESS ADMINISTRATION CRCM REVIEW

SBA currently operates 15 sensitive information systems. Several large systems are aggregates of other systems. For example, the Loan Accounting System (LAS) is an aggregate system which includes three major applications: (1) Loan Origination and Disbursement, (2) Loan Accounting and Collection Processing, and (3) Loan Servicing and Debt Collection. These applications include approximately 24 individual application subsystems.

ESTIMATED LEVEL OF EFFORT

In preparation for the CRCM review of LAS, it was necessary to determine the level of effort required to conduct the review. We anticipated that the review would be contracted out and performed by a qualified computer security contractor. We also estimated that approximately 100 staff days of effort would be required. This effort is summarized in figure 3. The Contractor actually completed the review in 75 staff days.

Level 1	HLRA		8
Level 2	SLRA		2
Level 3	VA	(3 * 10)	30
Level 4	SRAR	(3 * 15)	45
Level 5	SQACR	(3 * 5)	15

Total staff days:			100

Figure 3. Estimated Level of effort

Figure 4 provides an overview of the LAS hierarchical structure. Figure 5 provides a mapping of the CRCM detailed procedures to LAS.

DETAILED REVIEW PROCESS

The CRCM review included the following detailed procedures:

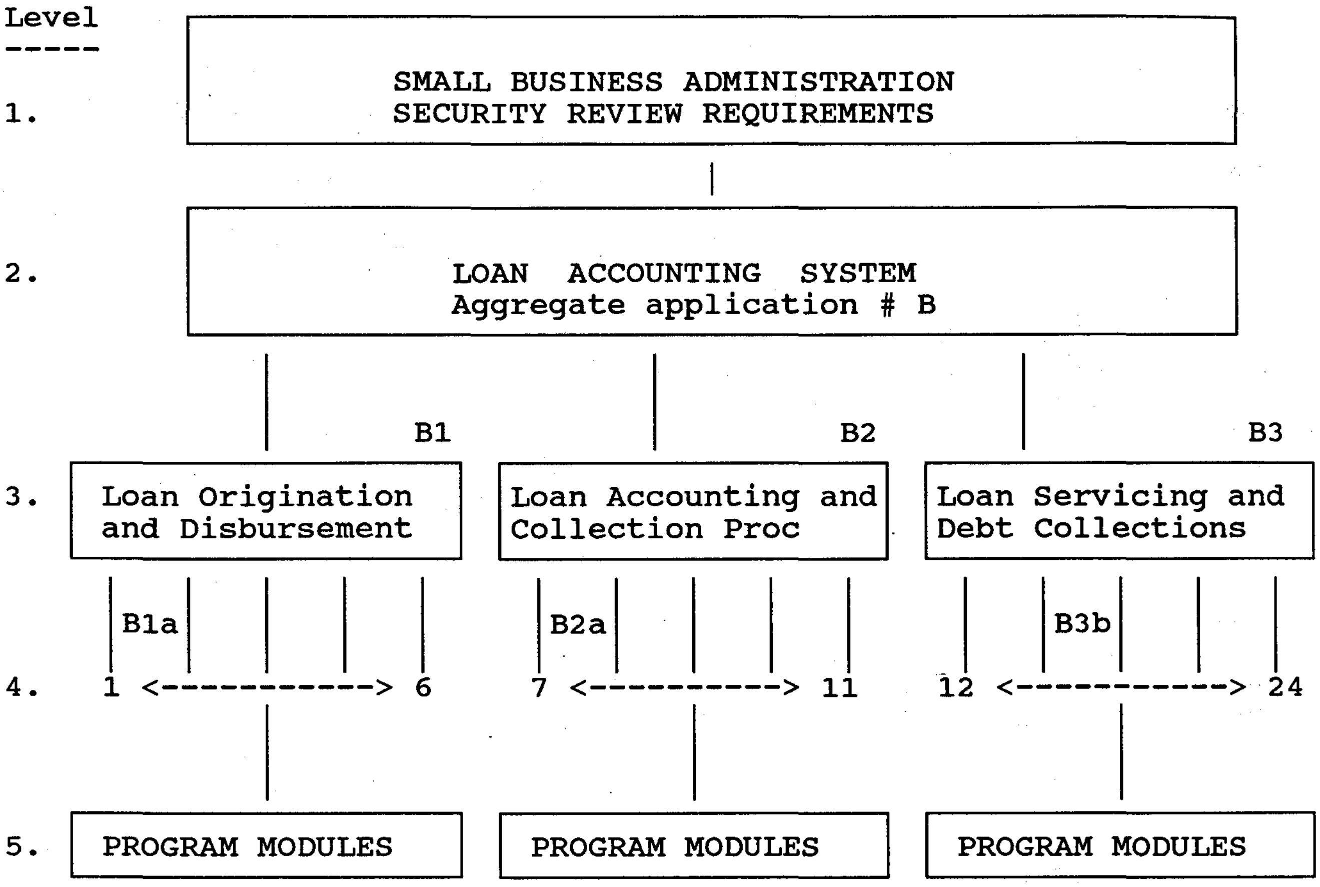
1. A high level risk assessment (HLRA) of the SBA 's computer security workload. This task included an update of the FY 89 HLRA using a qualitative risk assessment software. The results from this review included an inventory of the Agency's computer security workload for FY 90 in risk priority order.

2. An aggregate system level risk assessment (SLRA). The SLRA included an assessment of designated pervasive risks and special risks for each major system. Five pervasive risks and three special risks were assessed. The results of this review included an inventory, in risk order, of subsystems for each major system (see B1, B2, and B3 in figure 4). This inventory also identified the highest risk subsystem for each major system.

3. A vulnerability risk assessment (VRA) of major systems B1, B2, and B3. This assessment included the identification of vulnerabilities and installed control measures for each major system. An application control matrix was prepared for each major system using a software package. A review of vulnerabilities and of control objectives based on the PCIE Model Framework for Management Control was also conducted.

4. A security requirements analysis review (SRAR) of the highest risk subsystem from applications B1, B2, and B3 (subsystems B1a, B2a, and B3b). This review included a detailed evaluation of threats, risks, and controls. A Security and Integrity Requirements Worksheet was prepared according to SBA level 2 sensitivity. Controls were evaluated based on the requirements for SBA level 2 sensitive systems. The report from this review provided a detailed discussion of findings and recommendations.

5. A security and quality assurance compliance review (SQACR) of the most critical programs from high risk subsystems B1a, B2a, and B3b. This review included the evaluation of functional system requirements, application design standards, quality assurance standards, and security and control requirements at the program module level. A detailed Application Review Questionnaire was used as the basic data gathering tool. Program source code was also reviewed.


Level
1.
SMALL BUSINESS ADMINISTRATION
SECURITY REVIEW REQUIREMENTS
2.
LOAN ACCOUNTING SYSTEM
Aggregate application # B
B1
B2
B3
3.
Loan Origination and Disbursement
Loan Accounting and Collection Proc
Loan Servicing and Debt Collections
B1a
B2a
B3b
4.
1 <-----------> 6
7 <----------> 11
12 <-----------> 24
5.
PROGRAM MODULES
PROGRAM MODULES
PROGRAM MODULES

Selected the highest risk program modules from application subsystems B1a, B2a, and B3b.

Figure 4. Overview of the LAS hierarchical structure.

```
               -------------------------------
              |HIGH LEVEL RISK ASSESSMENT of  |
  --------    |computer security workload for |   Prioritized list of
 | CRCM-1 |-> |the Agency. Work units are     |-> security work units
  --------    |ranked according to pervasive  |   for the Agency
  (HLRA)      |risk areas and specific risk   |   according to risks
              |areas.                         |   ------------------
               -------------------------------      (APPENDIX A)

               -------------------------------
              |SYSTEM LEVEL RISK ASSESSMENT of|
  --------    |the Loan Accounting System.    |   Prioritized list
 | CRCM-2 |-> |Subordinate subsystems are:    |-> of LAS application
  --------    |1-Loan Origination & Disb      |   subsystems
  (SLRA)      |2-Loan Accounting & Coll Proc  |   according to risks
              |3-Loan Servicing & Debt Coll   |   -----------------
               -------------------------------      (APPENDIX B)

               -------------------------------
  --------    |VULNERABILITY RISK ASSESSMENT  |   Matrix of security
 | CRCM-3 |-> |using matrix analysis of:      |-> techniques installed
  --------    |1-Loan Origination & Disb      |   by control objectives
  (VRA)       |2-Loan Accounting & Coll Proc  |   ------------------
              |3-Loan Servicing & Debt Coll   |     (APPENDIX C)
               -------------------------------

               -------------------------------
              |SECURITY REQUIREMENTS ANALYSIS |
  --------    |REVIEW of high risk subsystems.|   Documented list
 | CRCM-4 |-> |Security & control techniques: |-> of controls based
  --------    |1-Loan Automated Allotment     |   on requirements for
  (SRAR)      |2-LACCS Daily Update           |   SBA leve 2 systems
              |3-Loan Accounting Online Update|   ------------------
               -------------------------------      (APPENDIX D)

               -------------------------------
              |SECURITY AND QUALITY ASSURANCE |   Specific findings
  --------    |COMPLIANCE REVIEW of critical  |   related tocompliance
 | CRCM-5 |-> |software.  The review focuses  |-> with SBA standards
 --------     |on High risk program modules   |   and QA requirements.
  (SQACR)     |identified in  CRCM-4.         |   ------------------
               -------------------------------      (APPENDIX E)
```

Figure 5. Mapping of CRCM review to LAS

CONCLUSION

The major benefit from the CRCM review was that SBA's Loan Accounting System was certified and accredited within a reasonable time frame and at reasonable cost. The CRCM procedure addressed the areas of high risk, and provided acceptable results to Agency certifying and accrediting officials. The results were used to make an informed decision on whether to accredit fully, to accredit with specific qualifications, or not to accredit the system.

Using current review procedures and traditional security review techniques, the resources to review and certify LAS would have required more than 300 staff days.

APPENDIX A

GENERAL RISK ANALYSIS - FY 90

```
RISK PRIORITIZATION REPORT         *     AVAILABLE WORKDAYS: 330
                                   *
   ACTIVITY                TOTAL   *   SENS                WORK  ACCUM
   NAME                    RISK   NORM IMPT TYPE SCOPE DAYS  DAYS
-------------------------  ----- ***** ---- ---- ----- ----  -----
PAYROLL USER INTERFACE     1175   .967  9   CER  COM   10    10
ADP SECURITY TRAINING      1160   .955  9   ADM  COM   10    20
ADP SECURITY PROG MGMT     1160   .955  9   ADM  OPR   120   140
8(a) FIN INFO SYSTEM       1138   .936  9   CER  COM   20    160
SURETY BOND GUARANTEE      1133   .932  9   CER  COM   20    180
LOAN ACCOUNTING SYSTEM     1130   .930  9   CER  COM   75    255
NEW SYSTEM COORDINATION    1128   .928  9   TA   OPR   10    265
DISASTER AREA CONTROL SYS  1125   .926  9   CER  COM   5     270
SALARY AND EXPENSE SYS     1123   .924  9   CER  COM   40    310
OFFICE OF FINANCIAL OPER   1113   .916  9   CER  COM   10    320
SBA NETWORK SECURITY SYS   1113   .916  9   CER  COM   10    330
       ****  Accum workdays = or > Available workdays ****
SNSS FUNCTIONAL REQMTS     1098   .903  9   TA   AIS   15    345
COMPUTER SEC ACT FOLLOWUP  1095   .901  9   ADM  COM   10    355
AGENCY PERSONNEL SYSTEM    1085   .893  9   CER  COM   10    365
ADP CONTROLS MANUAL        1068   .879  9   ADM  OPR   20    385
PROCUREMENT ASSISTANCE     1040   .856  9   CER  COM   10    395
PROCUREMENT AUTO SOURCE    1035   .852  9   CER  COM   10    405
PERSONNEL DATA SYSTEM      1033   .850  8   CER  COM   10    415
COMPUTER INTL CTL REV SYS  1018   .837  9   CER  COM   20    435
SPERRY MAINFRAME FACILITY  1013   .833  6   RA   COM   50    485
END USER ASSISTANCE        1010   .831  9   TA   OPR   50    535
REGIONAL INFO PROC CENTER  995    .819  9   CER  COM   5     540
BUS DEV MGMT INFO SYS      983    .809  9   CER  COM   10    550
SPERRY MAINFRAME COMM      968    .796  6   VA   COM   20    570
ADVOCACY INFORMATION SYS   945    .778  9   CER  COM   10    580
PRIME CONTRT REG INFO SYS  855    .704  9   CER  COM   5     585
WANG WORDPROCESSING CTR    800    .658  6   RA   COM   5     590
REGIONAL OFFICE END USERS  793    .652  6   RA   COM   10    600
CENTRAL OFFICE END USERS   760    .626  6   RA   COM   5     605
INFORMATION CENTER         748    .615  6   RA   COM   5     610
DISTRICT OFFICE END USERS  733    .603  6   RA   COM   10    620
PROC CAREER MGMT SYSTEM    588    .484  5   CER  COM   5     625

  TOTAL ASSESSABLE UNITS = 32      AVERAGE NORM SCORE =  0.827

                      QUALITATIVE RISK CONVERSION          1
                                                           0
 .......15...............45...............75........90.....0
                           L<-----M------>H
        /\                /\                 /\        /\
 -------  ---------------  ---------------   --------  ------
 Very Low      Low           Moderate          High     Very High
                                                 ^
     FINAL RISK RATING                   ==> 0.83   ‖
```

APPENDIX B

GENERAL RISK ANALYSIS - LAS

```
TYPE CONSOLIDATION REPORT         *     AVAILABLE WORKDAYS:
                                  *
   ACTIVITY                TOTAL  *  SENS                WORK  ACCUM
   NAME                    RISK  NORM IMPT TYPE SCOPE DAYS  DAYS
------------------------- ----- ***** ---- ---- ----- ----  -----
MAIL ACCT SYSTEM           635   .543   5  APP  ICR   15    15
MAPPER PILOT               610   .521   5  APP  ICR   15    30
                     AVERAGE =  0.532
LOAN ACCT OL UPDATE        1123  .959   9  AP3  CER   15    15
DELINQ LOAN COLLECTION     1028  .878   9  AP3  ICR   15    30
QUERY TRANSACTION          1025  .876   9  AP3  CER   15    45
SALARY OFFSET              1015  .868   8  AP3  ICR   15    60
DELINQ LOAN COLL REPORT    1003  .857   9  AP3  ICR   15    75
IRS OFFSET                 980   .838   8  AP3  ICR   15    90
COLLECTION AGENCY          873   .746   8  AP3  ICR   15    105
LIT/LIQ TRACKING           830   .709   7  AP3  ICR   15    120
TRANS APP DISP             773   .660   6  AP3  ICR   15    135
GUARANTEE LOAN REPORT      718   .613   5  AP3  ICR   15    150
8(a) FIN INFO SYSTEM       683   .583   5  AP3  ICR   15    165
NAME & ADDRESS SYSTEM      635   .543   5  AP3  ICR   15    180
CREDIT BUREAU REPORT       590   .504   4  AP3  ICR   15    195
                     AVERAGE =  0.741
GUARANTEE PURCHASE SYSTEM 1138   .972   9  AP2  CER   5     5
LOAN ACCT ALLOTMENT        1078  .921   9  AP2  CER   15    20
LOAN APP 327 MODIFICATION 1038   .887   8  AP2  CER   15    35
LOAN DISB. -1416           993   .848   9  AP2  CER   15    50
LOAN APPL. TRACK           938   .801   8  AP2  CER   15    65
POLK FILE PROCESSING       585   .500   4  AP2  ICR   15    80
                     AVERAGE =  0.822
LACCS DLY UPDATE           1113  .951   9  AP1  CER   15    15
LOAN ACCT REPORT           1088  .929   9  AP1  CER   15    30
MGT INFO SUMMARY           945   .808   9  AP1  CER   15    45
TREASURY (LAT)             728   .622   6  AP1  ICR   15    60
FED ASSIST AWARD           578   .494   4  AP1  ICR   15    75
                     AVERAGE =  0.761

  TOTAL ASSESSABLE UNITS = 26       AVERAGE NORM SCORE =  0.747

                     QUALITATIVE RISK CONVERSION           1
                                                           0
 .......15...............45...............75........90.....0
                          L<-----M------>H
        /\               /\                /\        /\
 -------  ---------------  ---------------  --------  ------
 Very Low       Low          Moderate          High    Very High
                                             ^
          FINAL RISK RATING      ==> 0.75    ‖
```

APPENDIX C

CONTROL MATRIX

Below is a sample matrix for an automated system. CONTROL OBJECTIVES are broad objectives which must be further codified depending on the sensitivity of the system. The number in each cell is the assessed rating for the effectiveness of the CONTROL TECHNIQUES in place.

OBJECTIVES	VULNERABILITIES/RISK				
	USER PROCEDURE	UNAUTH ACCESS	FILE/DATA MANIPULATION	FRAUD POTENTIAL	DESTRUCT OF DATA
AUDITABILITY	3 H	5 D,G	5 D,G	4 G	4 G
CONTROLLABILITY	5 C,H	5 A,B	5 C,F	N/A	N/A
DATA INTEGRITY	N/A	3 A	5 C,F	3 F	N/A
FUNCTIONALITY	3 E	3 E	3 E	4 E,F	3 E
IDENTIFICATION	3 E	3 H	4 E,F	4 E,F	3 E
SEPARATION OF DUTIES	3 F	3 F	3 F	3 F	N/A
RECOVERABILITY	3 E	4 G	2 D	N/A	4 G
SURVEILLANCE	3 F	4 G	2 D	3 H	3 C
SYSTEM DESIGN INTEGRITY	5 A,F	4 G	2 D	4 G	3 E
Vulnerability level:	3	4	3	3	2

Explanation of control technique codes:

(A)-A large number of edits and validation checks are included.
(B)-Data limit checks are used as prescribed by management.
(C)-User performs reconciliation of inputs to outputs.
(D)-Exception reports are produced for users.
(E)-Operating procedures are used, and are regularly updated.
(F)-Review of input preparation is provided by management.
(G)-Logs are maintained, such as: source document, terminal access, data base changes, program changes, and media storage.
(H)-Evidence of authorization is recorded and periodically reviewed.

APPENDIX D

SYSTEM SECURITY AND INTEGRITY REQUIREMENTS WORKSHEET

SYSTEM: LAS SENSITIVITY LEVEL = 2

This document specifies the security and integrity requirements of the LAS system. It also establishes specific requirements for the certification of the LAS system. The certification review process is undertaken to compare the system to these documented requirements. The certification of LAS is made relevant to its compliance or noncompliance to these stated requirements.

1. AUDITABILITY

Auditability permits relative ease in examining, verifying, or demonstrating a system. Auditability's two prime objectives are to:

- Determine whether the proposed system contains a network of internal controls adequate to ensure that the results of the system will be reasonably accurate and reliable; and
- Determine whether the planned management/audit trails will satisfy management's need for periodic inquiry and enable auditors to perform an effective audit.

2. CONTROLLABILITY

The objectives of internal control are to provide management with reasonable, but not absolute, assurance that financial and other resources are safeguarded from unauthorized use or dispositions; transactions are executed as authorized, financial and statistical records and reports are reliable; applicable laws, regulations and policies are adhered to; and resources are efficiently and effectively managed.

3. IDENTIFICATION

A key element in data security is identification, which enables an organization to hold authorized users accountable for their actions. To accurately identify users, security mechanisms need certain information concerning each attempted access.

4. INTEGRITY

Systems integrity is the property of a system that permits effective and reliable development and use; and normally requires a design methodology, structured development, and up-to-date operational procedures.

5. ISOLATION

Isolation protects the system and its data from unauthorized sources. This is achieved by segregating functions and duties to specific activities on a "need to know" or a "need to perform" basis.

6. RECOVERABILITY

Recoverability of data and processing includes emergency reaction capability, backup precautions, and the ability to fully recover any lost or damaged system resource. File reconstruction and system fall back procedures are major considerations for recoverability.

7. SURVEILLANCE

Surveillance enhances security by detecting potential threats and by accounting for all accesses and attempted processes by each user. It includes the recording of significant events to allow audit trails or monitoring components to detect potential or occurring breaches and raise an alarm.

8. FORMAL CONTROL OBJECTIVES

Formal control objectives refer to regulatory objectives that must be met. Regulatory objectives include security policy, control policy, accounting policy, and other regulatory requirements that are applicable.

9. SPECIFIC RISKS SECURITY AND CONTROLS

Three SPECIFIC RISKS were identified in the sensitivity assessment. Describe the approach used to control these risks or reduce the vulnerability and consequence of their occurrence. Address each risk by name.

10. FUNCTIONALITY

Functionality includes the minimum functions that must be performed by the system to be considered successful. These are basic user requirements that the system must meet.

11. EXCEPTIONS

Exceptions refer to constraints that are imposed upon the system due to financial or technical considerations. Of primary concern are those constraints that may have an impact on integrity and security of the system.

12. SIGNATURES

The undersigned provide their acceptance of the above items as a reasonable set of integrity and security requirements for LAS.

SECURITY REQUIREMENTS SELECTION

Security requirements consist of specific control measures necessary to protect designated control areas. The evaluation and certification of security requirements must be translated into a set of controls appropriate to the sensitivity of the system.

CONTROL OBJECTIVES	SENSITIVITY LEVEL			
	1	2	3	4
AUDITABILITY				
- Audit trails	x	x	x	
- Special audit features	x			
CONTROLLABILITY				
- Systems management	x	x	x	
- Project management	x	x	x	
- System development methodology	x	x		
- System documentation	x	x	x	
IDENTIFICATION				
- Passwords	x	x	x	
- Password administration	x	x	x	
INTEGRITY				
- Program standards	x	x		
- Data validation	x	x	x	
- Error reports	x	x		
- Application controls	x	x		
- Reconciliation procedures	x			
- System test	x	x	x	x
- Test plans	x	x		
ISOLATION				
- Segregation of activities	x	x		
- Documented system boundaries	x	x		
RECOVERABILITY				
- Data backup	x	x	x	x
- Data retention	x	x	x	x
- Contingency plan	x	x	x	
SURVEILLANCE				
- Enhanced audit trails	x	x		
- Error reports	x	x		
- User monitoring	x			
- Security test	x			

APPENDIX E

CERTIFICATION REPORT

The certification report is the primary product of the certification review. It contains a summary of security and integrity measures and technical security recommendations. It is the main basis for the accreditation decision. The certification report is prepared by the Agency Certification Program Manager.

1. EXECUTIVE SUMMARY

2. BACKGROUND

3. FINDINGS

    - Internal Control Posture
    - Systems Integrity Posture
    - Vulnerabilities
    - Risk Summary

4. RECOMMENDED CORRECTIVE ACTIONS

5. SUMMARY OF CERTIFICATION PROCESS

Attachments

A - Proposed Accreditation Statement
B - Security and Integrity Requirements Worksheet
C - Control Matrix Report
D - Application Contingency Plan
E - Computer Security Plan

APPENDIX F

EVALUATION KEY WORDS

CRITICAL RISK CERTIFICATION METHODOLOGY (CRCM): A security review and evaluation of risks of large aggregated sensitive applications at various hierarchical levels. The review process focuses on "critical risks" at each level. Each lower level is accorded a more detailed review than its parent level.

HIGH LEVEL RISK ASSESSMENT (HLRA): A qualitative risk assessment of Agency computer security workload. This assessment determines workload inventory and the measure of risk associated with each computer related activity in the inventory.

SYSTEM LEVEL RISK ASSESSMENT (SLRA): A qualitative risk assessment of the application within the aggregated application. Pervasive risks and special risks are assessed at the major system level in order to determine the high risk subsystems.

VULNERABILITY RISK ASSESSMENT (VRA): An assessment of vulnerabilities and security measures of major systems within the aggregated system. An application control matrix, which cross references vulnerabilities, control objectives, and security techniques, is developed. Detailed steps include the use of software for the assessment of vulnerable areas and control objectives based on the PCIE Model Framework for Management Control Over Automated Information Systems. As an alternative, the high risk subsystem may be assessed in lieu of the major system.

SECURITY REQUIREMENTS ANALYSIS REVIEW (SRAR): A detailed security review of the highest risk subsystem in each major system. The controls evaluated are determined by the sensitivity level of the major system.

SECURITY AND QUALITY ASSURANCE COMPLIANCE REVIEW (SQACR): A source code evaluation of the most critical program module from t e highest risk application identified above. This review includes the evaluation of design standards, software standards, functional requirements, and security, integrity, and control requirements.

GENERAL KEY WORDS

CONTROL: A policy, technique, practice, device, or programmed mechanism to accomplish an integrity or quality objective.

CONTROL OBJECTIVE: A positive condition or event to be obtained, given that there may be associated negative events operating to hinder or prevent the positive event from occurring.

COUNTER MEASURE: Protective action, device, procedure, technique or other mechanism which increases the difficulty of exploiting computer system vulnerabilities (also called SAFEGUARD). PERVASIVE RISK: General risks which are applicable to all organizational activities.

RISK ANALYSIS: The identification of threats, vulnerabilities, and the likelihood that the organization will experience some negative impact as a result of threat occurrence. May also include definition of specifics leading to the vulnerability, and the additional controls required to "manage" or if possible, eliminate threats.

RISK ASSESSMENT: A systematic method for analyzing specific computer activities, applications, vulnerabilities to establish potential loss or adverse impact to an organization from certain events based on estimated likelihood of occurrence of those events

SECURITY CERTIFICATION: A technical evaluation (security evaluation), made as part of and in support of the security accreditation process, that establishes the extent to which a particular computer system or application meets a prescribed set of security requirements.

SENSITIVITY: The degree of criticality of computer system components to their owners, users, or subjects and is most often established by evaluating the risk and magnitude of loss or harm that could result from improper operation or deliberate manipulation of components.

SPECIAL RISK: Inherent risks unique to a specific activity. Examples are: (1) potential for fraud, (2) lack of management control, and (3) potential for law suits.

VULNERABILITY: The motive, opportunity and means by which a threat can be actuated. This reflects present conditions within the organization which might lessen or intensify risk. It is also the susceptibility to loss, and condition that allows a loss to occur.

REFERENCES

[1] U.S. Small Business Administration, Automated Information Systems Security Program, SOP 90 47, August 1987.

[2] U.S. Small Business Administration, General Risk Assessment System (GRA/SYS) software User Manual.

[3] U.S. Small Business Administration, CONTROL MATRIX Software User Manual.

[4] C. H. Helsing, "Application Risk Assessment and Controls Selection", Datapro Reports on Information Security, Datapro Research Corporation, 1986, IS20-300-101.

[5] National Bureau of Standards, "Work Priority Scheme for EDP Audit and Computer Security Review", NBSIR 86-3386, Gaithersburg, Md.,1986.

[6] National Bureau of Standards, "Guide to Auditing for Controls and Security: A System Development Life Cycle Approach", Special Publication 500-153, Gaithersburg, Md., 1988

[7] J. Fitzgerald, Designing Controls Into Computerized Systems, Jerry Fitzgerald & Associates, 1981.

[8] National Bureau of Standards, "Guidelines for Security of Computer Applications", FIPS PUB 73, Gaithersburg, Md., June 1980.

[9] National Bureau of Standards, Guidelines for Computer Security Certification and Accreditation, FIPS PUB 102, Gaithersburg, Md., September 1983.

[10] National Bureau of Standards, "Report on the NBS Software Acceptance Test Workshop", Special Publication 500-146, Gaithersburg, Md., April 1986.

[11] U. S. General Accounting Office, Evaluating Internal Controls in Computer-Based Systems, Audit Guide, Washington, DC, June 1981.

[12] T. W. Osgood, "A Risk Analysis Model for the Military Environment", Procedings of the 11th National Computer Security Conference, October 1988.

[13] J. Stevens, "Managing the Accreditation Process: Lessons Learned", Proceedings of the 11th National Computer Security Conference, October 1988.

FACTORS EFFECTING THE AVAILABILITY OF SECURITY MEASURES

IN

DATA PROCESSING SYSTEM COMPONENTS

by

Robert H. Courtney, Jr.
Robert Courtney, Inc.
Box 836, Port Ewen, NY 12466

INTRODUCTION

With very few exceptions, our ability to provide adequate computer security is not limited by the availability of appropriate technology. We know how to make most of the hardware and software security controls needed now to make our systems far more secure than they are and adequately secure for most of today's operating environments. Yet many systems are less secure than they should be because the suppliers of data processing products have not made available to their customers those functional characteristics needed for adequate security at reasonable cost.

Although the vendors are not providing what is needed, they do not bear a major portion of the responsibility for that situation. Our criticism of the vendors is mitigated significantly by the often good, if not always sufficient, reasons they have for not providing the security measures their customers need. We address those reasons later here. On the other hand, it is fair to condemn some of them for their hypocritical claims to be doing far more about security than they are. For example, one major vendor proudly quotes key phrases from its corporate policy on providing in its products the security its customers need while being quite aware of gross security deficiencies in some of their newest and most important products.

A fairly complex array of factors serves to limit the availability of security measures which, by their absence, make our rapidly growing dependence on computer-based systems much less safe than it should be. It is our purpose here to examine some of the more prominent of these factors.

THE SECURITY PRODUCTS MARKET

The Customers' Data Processing Management as a Factor.

Since the data security issue first gained prominence in the middle 1960's, the security of data in computer-based systems usually has been the responsibility of those managing the major data processing functions for their respective organizations. This followed quite naturally the rather general assumption that data security was a problem which usually could be solved by greater physical security for the data center and the inclusion of appropriate technical measures in the hardware and software complex. We were slow to realize that few technical measures can be effective without complementary policies, procedures and practices for which the data processing management has limited, if any, direct responsibility.

The pressures on data processing management to bring new applications on-line and their desire to have the growing dependence on their systems seen in the best light by the corporate management have not been wholly compatible with the need to divert some resources to the protection of data. Once the corporate management has been sold on committing major resources to the development of new systems, there is often some reluctance to then tell them about the threats and vulnerabilities associated with those new systems if, indeed, the developers were aware of them.

In addition to a rather general desire not to make security problems unnecessarily prominent, the DP management often has been, and usually still is, quite unaware of most of the threats to and vulnerabilities of the data and of the economic consequences of encountering the full array of security problems to which their data are heir. Without that awareness, they often do not understand that the incorporation of security measures into their systems is a way of reducing costs otherwise imposed on them by human frailty in all its forms, including carelessness, incompetence, mistakes, avarice, and malice, and by the perversity of inanimate objects and Mother Nature. Without that awareness, there is little reason for them to want to divert resources to the provision of adequate security.

Clearly, if the DP management is not adequately aware of both the nature and the degree of the potential computer security problems in their particular environment, it must be expected that they will not be seriously perturbed about them.

It has been our experience that very rarely has the DP management properly identified the most significant data security problems for which they have at least some degree of responsibility. Under these circumstances, it is not surprising that the DP management, the primary customers in most organizations for data processing products, has not generated a meaningful, coherent marketplace for much-needed security measures as either stand-alone products or as functional attributes of other product offerings, including mainframes and minis, peripherals, system control programs, data base management systems, and application programs.

Those in charge of corporate information systems will often contend that they have implemented a number of important security measures and, for that reason, things are pretty well under control in their shops. Such is usually not the case.

All too often those measures are not fully installed or fully utilized or have been compromised by the procedures which surround them.

The most commonly installed ones are those with high cosmetic value, but which may have very limited security value in the absence of other key measures which are not in place. There are numerous instances where such products have been bought at quite significant cost and then never used.

The Major Vendors as a Factor.

The most successful vendors of data processing products did not achieve that position by responding to customer demands. Of necessity, they anticipated customer demands or, even more frequently, generated customer desire for new products. They made their products available and then explained to their customers why they needed them.

Product development lead times are not conducive to response to customer demands other than for limited modifications to already available products. Customer interest can wane or be diverted to other things before the once-desired product can be made available. Success dictates that the vendor generally anticipate and not just respond to customer demands. The question, then, is whether the potential customers want or can be induced to want security measures if they are made available.

It is often quite difficult to sell security-related products or added-cost security features on existing products. Salespersons are rarely interested in selling a product which requires that they spend much time motivating or educating the customer if they can sell, with the same or less effort, another product which is equally, if not more, profitable for them. Further, it is not wholly reasonable to expect salespersons to sell a major system and then go to any significant trouble to identify and tell the customer about the potential dangers, the security problems, which he must anticipate when using that product.

As things are today, we cannot expect the major vendors to do a thorough job of making available in their products the security attributes needed to make major, complex systems adequately secure for most applications until there is a market for those attributes which is about as profitable as would be the added function and performance which could be included in a product at the same cost to the vendor. Today's market for security measures, other than that highly tenuous federal market generated by the activities of the DoD's National Computer Security Center, does not provide adequate motivation for the major vendors to mount other than severely limited security product programs.

Even when a major vendor introduces security products those products, these products are often components of systems which clearly demand security to be saleable, such as ATM network components or an EFT system, with the result that those security measures which are provided lack the generality needed to satisfy customers using those components in other ways. It is often true that generality could have been included had the product managers been able to see a greater potential market for those security attributes.

This is a reflection of the generally low level of awareness of customer security needs by major vendor product developers which, in turn, reflects a generally negative attitude on the part of vendor management about the market for security measures.

Customer Organization as a Factor.-
In earlier paragraphs here, it was said that a major security problem is presented by the lack of awareness on the part of most DP management of the quantitative aspects of security and a consequent failure to recognize that all properly selected security measures, by definition, will displace direct or indirect costs greater than the cost of those measures. Because of this situation, it is appropriate to consider alternative assignments of security responsibility to bring to bear on the problem the concerns of those in better positions to recognize the need for security, to assess the consequences of not having it, and to provide data to support cost-benefit determinations in the selection of appropriate security measures.

The assignment of primary responsibility for data security to the managers supported by data processing rather than its assignment to the data processing organization can have highly salutary effect on an organization's computer security posture. If that responsibility is given to the management of the respective functional areas of the organization, whether it be in the public or private sectors, and if the management is directed to evaluate both the direct and indirect costs of accidental or intentional modification, destruction, disclosure, delay or loss of availability of the data supporting their functions, then there can be a basis for determining what security measures are needed and the cost justification for each.

Once there is firmly assigned responsibility and a mechanism for the selection and cost justification of security measures there is a basis for the procurement of that, and only that, security which is needed. Until such changes are made in the assignment of responsibility in a substantial number of organizations, it is improbable that we will see the evolution of a rational market for security measures and the subsequent provision of those measures by the vendors.

We do not suggest that those changed assignments of responsibility will alone create a market but rather that they are a necessary if not sufficient condition for its evolution. There are other significant factors which are treated below.

A full discussion of the recommended organization is somewhat beyond the scope of this paper, but the matter is discussed in detail in R. Courtney, The Proper Assignment of Responsibility for Computer Security, Computers and Security, Volume 7, No. 1, Feb., 1988.

The Small Vendor as a Factor.

The extensive publicity given computer security issues in the general and trade press has led to the development of a fairly large number of security products in both hardware and software. Most of these are offered by small vendors many of whom have only that single product or, at best, a few minor variations on the same basic technology. Roughly one in every five such products are likely to provide a cost-effective, or even just effective, means of containing the problems for which they were developed or address problems which are of significant importance to warrant enough customer interest to constitute a meaningful market.

Several of these products are potentially quite useful, but most are proving too hard to sell. The salesperson for a single product has a very difficult time unless he can get within a reasonable time orders for quantities large enough to justify the sales costs. Many persons selling such products get only orders for quantities of one and two for evaluation. That evaluation process is often long or never happens or produces negative results. The sales costs then become too large a portion of all expenses and the product or the whole vendor company then fails.

There is not yet a recognized marketing organization, a manufacturers' rep, selling a meaningful variety of such products for a substantial number of vendor companies so that the salespersons can afford the time to develop adequate familiarity with the potential customers' needs and their people that they can have reasonable expectation of getting significant orders for at least some of the many products they should be offering. Until such an organization, manned by people with real expertise in the field, does evolve, the probability that a very small security product vendor, even one with a very good product, will survive is discouragingly low.

The National Computer Security Center (NCSC) as a Factor.

The NCSC, an organization within the National Security Agency (NSA) which is, in turn, part of the Department of Defense, was established to address computer security as a means of protecting information involving the national security. Their activities became highly politicized as a consequence a National Security Decision Directive #145 (NSDD #145) issued by President Reagan in 1984. The effectiveness and credibility of the NSDD were not enhanced by an attempt by Admiral Poindexter to clarify its purpose and meaning three weeks before he was deposed as a consequence of his involvement in the Iran-Contra affair.

In late 1984, not long after NSDD#145 was issued, the Director of NSA announced, in a talk to an IEEE group in Washington, D.C., the intention of NSA/NCSC to convince the civil agencies of government and the private sector that their needs for computer security measures were the same as those of agencies trying to protect multilevel classified data from highly motivated technical giants in the employ of major national enemies. His stated purpose in doing that was to induce these other organizations to use the measures being specified by the NCSC and, as a consequence of the increased volume, reduce the cost to the government of the security products that NSA wanted it to have.

Unfortunately, the NCSC criteria completely ignored data integrity considerations and centered exclusively about protection against violations of confidentiality by people wholly outside of the organization. In all reality, of course, it would be unwise indeed for many organizations to ignore the timeliness, accuracy, completeness and availability of data while converging their concerns solely on maintenance of confidentiality. For this reason, the NCSC criteria, which have attracted widespread attention as a consequence of very aggressive missionary work by the NCSC, precipitated interest far beyond their significance in even most of DoD as well as the civil agencies of government and the private sector.

The cause of data integrity would have been better served if only half of the funds used by the NCSC in overselling their confidentiality criteria had been applied to the development of integrity criteria.

Because the DoD perceived in NSDD #145 enough leverage to do so, the civil agencies of government were ordered to be in compliance with certain of the NCSC criteria by 1992. As a result, the major main frame and control program vendors were threatened with being locked out of the large federal market for main frames unless they rapidly brought their mainframes and control programs into compliance with the NCSC criteria. Thus, even though many vendors did not necessarily believe in the real need of their customers for equipment meeting the NCSC criteria, the demands of the marketplace generated by the "C2 by '92" requirement effectively forced the vendors to invest very large sums in the development and subsequent evaluation by the NCSC of their complying products.

The U.S. Congress, seeing potential abuses in the DoD's interpretation of NSDD #145, passed the Computer Security Act of 1987. This new law more narrowly defined the role of DoD and NSA in the computer security area and resulted in the withdrawal of the NSDD by the President's National Security Advisor. That new law also gave to the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), responsibility for developing and promulgating computer security information to federal agencies processing unclassified data.

The Computer Security Act also ended the requirement that civil agencies meet NCSC criteria - but that simply created yet another new and different problem. By the time the Computer Security Act was passed, a number of DP managers in the civil agencies had realized that they could simply procure a computer meeting the lowest significant NCSC criteria, C2, and, even though they did nothing else about security and even though the NCSC criteria are woefully inadequate in addressing data integrity and availability, ignore misconduct on the part of an agency's own people, and are completely silent on employee awareness and organizational policies, procedures, personnel security and contingency planning, they can claim to their management that they have actually exceeded their security requirements. In the meantime, they are woefully insecure.

The confusion and obfuscation created by the NSDD and its subsequent withdrawal has further served to confuse the marketplace for security products. The current situation in the federal agencies is such as to lend only disruption to that market.

There is further mention of NSA as a factor in the discussion later here of the effects of export restrictions and national security standards on the security products market.

The Office of Management and Budget (OMB) as a Factor.
Positive direction by OMB might make a real difference in the security posture of particularly the civil agencies, but it seems not to be forthcoming. Although OMB has the power and the demonstrated competence to provide strong, positive direction, they seem overendowed with an odd combination of timidity and lassitude. Thus, it appears that we cannot look to DoD agencies or to OMB for help in promoting an orderly, constructive market for security products in support of the needs of most of DoD, the civil agencies and the private sector.

The National Institute of Standards and Technology as a Factor.
Earlier here we noted the responsibilities given, actually returned, to NIST by the Computer Security Act of 1987. Starting in the early 1970s, NIST, then the National Bureau of Standards, moved quickly and well to formulate guidance in computer security for the federal agencies.

Well before the initiation of security work there, NIST people had extensive experience in dealing with other agencies and the private sector through their role in standards. They were well aware that the easiest way to get broad agreement on a proposition is to have most potential disputants participate in formulating that which is to be agreed upon. (This approach is notably different from the wholly unilateral, somewhat dictatorial one taken by NSA.) As a consequence of that experience and partly because of limited resources, working groups were formed with members from NIST, DoD, the civil agencies and the private sector.

Over a relatively short time the NIST groups produced a set of publications providing guidance in computer security on a highly diverse array of topics. The general acceptance of these documents was quite high.

The current federal Data Encryption Standard (DES) was a product of

NIST in the late 1970's.

The Carter administration effectively ended many cooperative endeavors between NIST and the private sector. The intent of the restrictions was laudable, to limit undue or unfair influence by certain segments of the private sector to the detriment of those not participating or not exerting equal influence, but those restrictions served to destroy numerous constructive joint programs, which was highly regrettable.

When the Computer Security Initiative was established first under the Advanced Research Projects Agency and later moved to NSA and, through a succession of titles, became the NCSC, the computer security budget in NIST became minuscule in comparison. There was a highly prevalent and highly unfortunate belief that the NIST security activities were somewhat superfluous in light of the extensive program plans and impressive budget of the NCSC.

Few persons realized that the scope of the NCSC program was limited almost wholly to protection of data against compromise by persons outside of the system and had nothing to do with the integrity of those data. As a consequence, the whole data security program in the federal government suffered badly from neglect of those aspects of security generally far more important than those being emphasized by the NCSC.

With the passage of the Computer Security Act in December, 1987, steps were taken to significantly augment the NIST security budget. It seemed as late as 3Q89 that, with the new budget, NIST was to be off again and running in the security area and would again be in a position from which they could make meaningful contributions to the establishment of reasonable standards and procedures for federal agencies and help to stabilize the security products market for the private sector. But that was not to be.

While there is no doubt that our form of government is the best yet devised, there can also be no doubt that it has some pretty gross imperfections. One of them is ability of senior, and therefore influential, congressional representatives to make mistakes with far-reaching and virtually unrectifiable, even if disastrous, consequences. Last November, in the closing days of the first session of the 100th Congress, a Congressman with considerable seniority in the House Appropriations Committee confused the National Science Foundation, against which he had a grudge, with NIST and seriously impaired the NIST budget. This, in turn, seriously impaired the NIST computer security program.

This loss of NIST support is important to the availability of security measures in the marketplace because NIST is the logical source of specifications for such measures. One of the principal NIST goals was the generation of such material which could be referenced in procurement documents and, as a consequence of appearing repeatedly in such documents, help define a market for products with those functional and performance characteristics.

The weakening of the NIST capability is a significant set-back for computer security in both the private and public sectors. Their personnel complement is severely limited and they are not as strong as they should be but they do have an important charter. Many organizations in both the public and private sectors will find that they benefit by assisting NIST in the computer security area through participation in their Computer and Telecommunications Security Council and its associated working groups. That Council provides a very important means for making the right things happen if the federal agencies, state and local governments, and the private sector maintain or expand their current level of participation.

Export Controls as a Factor.

The export controls imposed on data processing products as a function of their security characteristics are often arbitrary and capricious. There are good and sufficient reasons for denying to those in some foreign countries access to certain technologies including the means to make it difficult for us to conduct our intelligence operations - provided the cost to us through lost exports is not so high as to outweigh the real advantages provided by export controls.

It is wrong to make a major economic sacrifice for a minor or only imaginary intelligence gain. For this reason, there is a clear need for the application of well-informed value judgment and to the decisions to allow or deny exportability to specific products.

Even when a vendor recognizes and is motivated to satisfy the security needs of its domestic U. S. customers by the inclusion of appropriate security measures into its products, there is a distinct possibility that the product then might not be exportable because it might be seen as helpful to extra-U.S. organizations in protecting their data against our acquisition of them. Under those circumstances, the cautious vendor might well omit the security measures to avoid jeopardizing the salability of his product outside of the U.S.

Because half of the revenue and more than half of the net of some U.S-based vendors are from outside the domestic U.S. market, this effort to satisfy the security needs of its domestic customers could result in a halving of its market for a product. The justification for such actions should be beyond question and not left to that breed of bureaucrat whose every reaction is to block shipment just in case it causes problems for which they might be held accountable.

The well-known Orange Book issued by NSA/NCSC provides criteria against which base system components can be evaluated and ranked in accordance with the degree of security provided against unauthorized disclosure (only). NCSC also evaluates products and places those passing on an Evaluated Products List (EPL). The ranking ranges from "D" for a vanilla system with no particular security characteristics, through C1 and C2 to B1, B2, and B3 and on to the highest level, A1.

It is probably true that major system control programs, if initially designed to do so, could meet the B3 criteria without great difficulty. The modification of existing ones to realize that level is probably possible, but that has not been demonstrated. In either case, the evaluation program would be very costly and time consuming. It has been reliably estimated that the evaluation process alone increases the cost of at least some products by more than 50%.

IBM has announced that it will produce a B3 system control program. That may be a serious mistake because B3 products are not exportable. In fact, to add confusion, products "containing B3 function" are not exportable. B3 function is not defined anywhere. Seemingly, anything needed to make a workable B3 product, including the ALU, would be a B3 function. The wording does not say "functions unique to B3 or higher products", whatever they might be.

There is a distinct possibility that a vendor producing a good, stable product will inadvertently make a B3-capable product, even though it has not been evaluated as such, and find that it has now denied itself a major extra-U.S. market.

One of the least understandable of the export problems involving security measures is found with products incorporating the Federal Data Encryption Standard algorithm - DES. DES is not a secret algorithm. It is described in complete detail in documents available world-wide, including everything from Time-Life books to the federal standard publication through which it was initially promulgated.

Products employing the DES algorithm can be exported to foreign but U.S.-owned financial institutions. For example, they can be exported to the office of a U.S. bank in Frankfurt but not to a Ford Motor Company facility in the same city. In both the devices would be under the control of foreign nationals.

There are about 32 European manufacturers of DES chips or products employing the DES algorithm, which has now achieved the status of a de facto international standard. Yet limitations placed on the export of devices using DES deny U.S. suppliers access to major international markets for such products at a time when every reasonable step should be taken to improve our balance of trade.

Any arbitrary, capricious or unpredictable application of export controls on security attributes in data processing products must serve to discourage U.S. vendors from providing such measures in their products. Yet such security characteristics are needed to instill customer confidence in steadily increasing dependence on modern, efficient data processing systems and thus help to assure the acceptability of such systems.

Limiting the exportability of any desirable functional or performance characteristic limits the market and thereby limits the incentive for vendors to include it in the product design. Anything which discourages growth in dependence on computer-based systems will deny the affected enterprises opportunities to reduce operating costs. Anything which denies a U.S. business the ability to reduce operating costs will serve to diminish its competitive posture in the world marketplace. For these reasons, any ill-conceived export controls imposed by U.S. government agencies on information system components offer a real potential for reducing the competitive posture of U.S. businesses.

A rational, understandable, consistent, and stable national policy on export controls is sorely needed and well before we do further irreparable damage to U.S. industry as a consequence of the continued application of the irrational hodge-podge of controls being somewhat whimsically applied today.

A Diversity of National Security Standards as a Factor.

NSA's National Computer Security Center evaluates certain data processing system components to assess their relative strength in contributing to the effectiveness of the so-called "trusted computer base". Soon after the initiation of the NSA/NCSC security evaluation program, the Director of the NCSC announced that his organization would not evaluate any foreign security products offering capabilities provided by U.S.-made products. Whatever his actual intent in excluding those foreign products, it was assumed by many at the time that his intent was to keep foreign products out of the U.S. market. The significance of this, whatever the intent might have been, was immediately apparent to many European vendors.

Before the NCSC announcement of the exclusion of foreign products from their evaluation process, there was already rather widespread interest in western European countries for the development of their own counterparts to the NCSC Orange Book. Part of this was simply nationalism at work, but the clear deficiencies in the Orange Book contributed significant additional incentive to the generation of their own national computer security standards. Most particularly, the absence from the NSA document of any indication of concern for the quality of data was such an obvious and serious omission that it alone would justify a more realistic approach to the generation of a computer security standard by almost any developed country.

The attempt at the exclusion of their products from the U.S. market by a U.S. intelligence agency provided considerable added incentive for each of several major European countries to go their own way in the generation of national security standards. That development clearly jeopardizes the U.S. vendors' market for data processing products because it threatens a very serious fractionation of the very large portion of that market which has been held by the U.S. It threatens U.S. vendors with a requirement that they produce as many versions of their product as may be demanded by the differences in individual country security standards - many of which standards, as we noted above, were provoked by ill-advised actions of NSA in precluding foreign data processing products from the U.S. market.

It is apparent that both the U.S. government and industry should support aggressively the generation and broad acceptance of international security standards which would permit the continued free flow of our data processing products to the rest of the world. If we do not do that, country-peculiar standards will be used to lock our products out of the foreign markets just as the NCSC attempted to do to products originating from without the U.S.

The National Computer Systems Security and Privacy Advisory Board (CSSPAB) as a Factor.

The CSSPAB was established under the Computer Security Act. Its activities have been underway only since Spring, 1989. Recognizing that it takes some time for a fairly large group of people not accustomed to working together to sort themselves out and mount a meaningful program, their accomplishments to date have been modest indeed. More aggressive action by that board is urgently needed.

The problem of NIST funding has served to undo some constructive work which had been done by the Advisory Board. While it was reviewing the NIST work plan for the coming year and making what seemed to be constructive recommendations for changes to address better the needs of its constituency, the NIST budget was fragmented, as noted above, which rendered moot many of the board's informal recommendations to NIST.

It seems highly apparent that more aggressive stances on key issues, are needed if the Advisory Board is to make the contributions so badly needed of it.

In the meantime, those trying to secure information systems are often burdened by the unavailability of hardware and software security controls they need to do that job. Unfortunately, we are continuing to see the appearance of security products which are reasonably priced with relation to their manufacturing costs but which are still far too costly to yield appropriate cost-benefit relationships in any sibstantial number of applications. Technical excellance is not a good measure of applicability, but too few entrepreneurs in the computer security area seem to grasp that reality.

END

INTEGRATING COMPUTER SECURITY AND SOFTWARE SAFETY IN THE LIFE CYCLE OF AIR FORCE SYSTEMS

Albert C. Hoheb
The Aerospace Corporation
2350 East El Segundo Boulevard
El Segundo, California 90245–4691

ABSTRACT

Computer security and software safety are two specialized areas in which acquisition and approval are governed by different requirements and organizations within the Department of Defense (DoD). The system software built for Air Force systems must follow a series of sometimes conflicting standards both during development and after delivery. Despite these conflicting standards, computer security and software safety are two disciplines with often similar requirements and goals. Rather than treating these subdisciplines separately, this paper suggests integrating them in the software acquisition and accreditation processes.

Currently in the development of system software, computer security and software safety are not integrated with each other; neither are the acquisition and approval of systems integrated together. Compounding this problem is an inherent difficulty in communications between organ-izations charged with procurement, implementation, and use, both among the services and between the services and their contractors. This complicates the design, delivery, and utility of safe and secure computing systems.

This paper presents the similarities and differences between computer security and software safety, reviews how software contributes to hazards, discusses how criticality relates to safety and mission requirements, suggests how acquisition and accreditation can be coordinated, describes how risk factors influence system design, and closes with how to implement software safety using computer security mechanisms.

INTRODUCTION

Recently, the computer security threat of denial of service was dramatically demonstrated when the Internet worm brought the network to a halt, spawning a vigorous interest in computer security [26]. The contributions of poorly designed or "unsafe" software to mishaps have not only been recognized, but they are also becoming a focus of attention at administrative levels.

Both computer security and software safety mechanisms should be considered to be *trusted* mechanisms, because they provide required services while preventing inadvertent, unintended, and malicious actions. Within the DoD, the Air Force is placing an increased awareness on concept definition, design requirements, and computer security and software safety mechanisms. Heretofore these features have been integrated within the acquisition process or the accreditation process; these two processes themselves have not yet been integrated with each other. This paper proposes just such an integration.

The commonality of computer security and software safety has previously been given little attention. The common design features and assurances used to mitigate risks have not been well understood, nor have their relationships to the acquisition and accreditation processes themselves [27, 12]. As software plays an ever larger role in systems control and risk in Air Force systems, computer security and software safety play correspondingly larger roles throughout the life cycle of computer-dependent systems. The trend has been to carve out computer security (COMPUSEC) and software safety as separate disciplines and to shift responsibilities away from security engineering and system-safety engineering organizations.

Because computer security and software safety have common irreducible elements, in particular common requirements and assurances, this paper proposes a merger of computer security and software safety in the unified discipline of *trusted computing.*

> This paper proposes the integration of the requirements of these two subdisciplines; furthermore, this paper proposes the integration of the acquisition and accreditation processes for critical trusted systems, as well as of the organizations committed to developing and maintaining those systems.

For simplicity, examples and specific recommendations have been limited to Air Force applications using Air Force requirements documents and terms; however, the integration concepts are applicable throughout DoD and even to civil and public sector systems.

COMMONALITY OF COMPUTER SECURITY AND SOFTWARE SAFETY

Technological breakthroughs in computing power, networking, and human interfaces have caused an increase in the use of computer controls, resulting in a corresponding increase in mission risks and program development risks. As the dependence of systems on computers has grown, so too has the demand for systems that are both secure and safe.

Computer security provides controls to preclude the unwanted disclosure of information or alteration of data, and to ensure that proper service is available when needed. Software safety provides controls to prevent data corruption which results in service interrupts, or the loss of life or assets. In both computer security and software safety, these controls are designed to guard against the risk of inadvertent, intentional, and unintentional threats.

It is not hard to find examples to illustrate the relationship between COMPUSEC and software safety. Suppose that a database is used in the performance of a system action whose improper execution could cause a hazard. This is the safety component. Now suppose that the security of the database is violated and some values are changed. Subsequent computer processing could then be both a security problem and a hazard. Thus, through this simple scenario, a cause-and-effect relationship between COMPUSEC and software safety can be seen. This type of relationship has led to the development of system requirements that cover both the security and safety aspects of Automatic Data Processing Systems (ADPS). For instance, Air Force regulation AFR205-16 [2] specifies five criticality factor (CF) levels (see Table 1). Criticality is defined within a specific context [4, 5], and is the required level of protection of resources, whose compromise, alteration, destruction, or failure to meet objectives will jeopardize mission accomplishment. For each CF level there is both a mission requirement and a safety requirement.

The relative priorities of the mission, security, and safety should be established so that each function is successful. For instance, in a satellite communications system the satellite's primary mission must be to communicate. As its secondary mission, the system must communicate securely; it also must be safe during all mission phases, especially during checkout and delivery, where there is a risk of human injury. In prioritizing the secondary mission requirements, one must consider that these requirements change throughout the mission. For example, initially it may be necessary to ensure that a satellite is extremely safe during checkout and delivery; later, during the operational mission phase, it may become critical to provide secure communications. Thus, if a failure occurs in that satellite's communication system, the requirements may demand that it communicate in an unsecure fashion rather than shut down completely.

The priority of various mission requirements results in a definition of criticality for each mission function. The criticality of each mission component then determines the balance of mission, security, and safety requirements. The program development risks must also be balanced, to ensure that the program can function as desired, within schedule and within budget. To the question "How safe and how secure does the system need to be?" there is no easy answer. Iterative analyses must be performed in order to balance requirements and minimize program development risk.

Table 1 compares computer security to software safety. Although aspects of these concerns change, both safety and security remain important through all phases of the mission. Both security and safety require attention from the conceptual design phase through operations and maintenance, yet this attention varies from predeployment checkout, through launch and all mission phases, to recovery. Vulnerability to exploitation results from the combination of threat and risk (for security) and hazards and risk (for safety). Both security and safety are vulnerable to the threat or hazard severity, and the likelihood that a threat or hazard will occur.

Computer security and software safety use design features and design and development assurances to eliminate or reduce the risks associated with specific threats and hazards. As Table 1 shows, the *foundation* for specifying computer security is to enforce policies of sensitivity, integrity, and service assurance [16, 4, 11]. Service assurance and integrity are the foundation of criticality [4]. This paper proposes that integrity and service assurance also be made the foundations of software safety, since integrity prevents modification and criticality provides the "need-to-do" (availability). In the

Table 1. Computer Security and Software Safety Elements.

Topic	Computer Security	Software Safety
Life cycle	All phases	All phases
Operations	All phases	All phases
Vulnerability	Threats	Hazards
Foundation	Service Assurance, Integrity, Sensitivity	Service Assurance, Integrity
Metric	Criticality Factors 1-5, Classes D-A	Criticality Factors 1-5
Control goals: to prevent	Inadvertent, Unintended, Malicious	Inadvertent, Unintended, Malicious
Objective: to prevent	Denial-of-service, Modification, Disclosure	Denial-of-service, Modification
Basis of Trust	Availability, Trustworthiness	Availability
Trust Principle	Least privilege	Least privilege
Sensitivity Levels:	Unclassified, Sensitive, Confidential, Secret, Top Secret	None identified

future, software safety levels based on integrity and service assurance policies may more clearly define software safety features and assurances.

The *metric* for computer security based on sensitivity [16] establishes classes of systems, from class D, the lowest assurance, to class A, the highest. These four classes have very specific design, mechanism, and assurance requirements, all of which increase in quantity and complexity as the metric approaches or exceeds class-A requirements. The metric for software safety, although separated into classes according to criticality factors, does not have specific design, mechanism, and assurance requirements.

The *control goals* are similar between computer security and software safety: to prevent *inadvertent, unintentional*, and *malicious* control action. *Inadvertent* "exploitation" (exploitation is commonly meant to mean the realization of risk) occurs when a control action is mistakenly invoked, e.g. when a misplaced coffee cup presses a key that in turn deletes a file. *Unintentional* exploitation occurs when a control action is purposely invoked but has an undesired consequence, e.g. when a component fails and the user, unaware of the failure, invokes a control action that in turn results in a hazard. *Malicious* exploitation occurs when an adversary, internal or external, threatens the system.

Both computer security and software safety share the *objective* of preventing denial of service, or the loss of system use. Only computer security has the direct objectives of preventing the unwanted disclosure of data and of preventing the unwanted modification of data through their alteration or destruction. The modification of data is only an indirect threat to computer security and software safety, since it is actually the processing of modified data that can directly result in denial of service, loss of assets, environmental damage, and injury or death.

The *basis of trust* (rationale) and *trust principle* for computer security are based on "least privilege" and assignment of roles, i.e., the granting of least amount of access according to user responsibilities. Availability ensures that the correct resources are provided at the correct time, Trustworthiness ensures that the resources operate correctly.

Both the computer security and software safety fields are struggling to define distinct policies based on use: policies specific to autonomous executive systems, embedded systems, stand-alone systems, and networked systems. Although computer security policies are better defined than are policies for software safety, software safety policies are established by users on a case-by-case basis.

Note that the Trusted Computer Security Evaluation Criteria (TCSEC) [16] has established assurance levels with regard to sensitivity, whereas to date no levels of software safety with regard to sensitivity have even been proposed. Nor has the TCSEC defined levels for integrity or service assurance.

As shown, computer security and software safety have a common operations and life cycle, as well as similar vulnerabilities, control goals, access basis and application, design goals, and foundation. The duality of these disciplines not only suggests but also encourages integration. Integration may be in the form of common requirements, common mechanisms, and common assurances.

COMPARISON OF COMPUTER SECURITY THREATS VERSUS SOFTWARE SAFETY HAZARDS

It is useful to decompose threats and hazards and then compare them to understand their commonality and other relationships. The COMPUSEC threat and software safety hazards are compared in Table 2 (this table is based upon the work of P. Neumann, and describes computer security threats [24, 3] but does not address common software development errors). Direct impacts to software safety are denial of service, loss of assets, environmental damage, and injury or death. All threats to integrity are direct threats, since execution of modified data does result, by definition, in a hazard.

Table 2. Comparison of Threats vs Hazards

	Software Contributions to Hazards					
Computer Security Threats	**Data Integrity**	**Commands**	**Interfaces**	**Hardware**	**Environment**	**Computer Processing**
Scavenging						
Traffic Analyses						
Elec. Interference	1	1	1	1	2	
Data Modification	1			2		
Hardware Tampering			1	1		
Jamming		1	1		2	
Communication Delay		1	1			
Impersonation		2				
Spoofing		2				
Worms, Virus, Logic Bombs	1	2				1
Data Hiding	1					1
Password Guessing		2				
Anthropomorphic Spoof		2				
Usage Flaws		1				1
User Privilege Alteration						2
Data Alteration	1					
Passive Misuse						
Covert Channels	1					
Misuse from Inaction		1				1
Message Stream Mods		1				

Indirect impacts are those that contribute to direct ones. The COMPUSEC threats encompass service assurance and attacks against sensitivity. For the most part, the software safety hazards correspond to attacks against integrity and service assurance. As we have seen, computer security threats contribute greatly to software hazards, both directly and indirectly.

As Table 2 indicates, hazards in command interpretation software can be triggered by command interference, jamming, communication delay, usage flaws, misuse from inaction, and message-stream modification, all of which result in denial of service. Impersonation, password guessing, and anthropomorphic spoofing are indirect threats to safety.

Generally, hardware threats are direct denial-of-service attacks. The contributions of environmental effects to safety have a secondary impact (unless, of course, they are created by a nuclear weapon). User privilege alteration affects safety if it permits inadvertent, unintentional, or malicious acts. The sensitivity attacks of scavenging, traffic analysis, passive misuse, and covert channels are associated only with the disclosure of data, and have no effect on software safety *per se*. Traffic analysis and passive misuse, although computer-related concerns, are traditionally classified as information security and TEMPEST concerns.

INTEGRATING THE ACQUISITION PROCESS

Traditionally, software has been developed along a "waterfall" model in which the next step cannot proceed until the current step is completed [20, 22, 21]. This model, although straight-forward, does not easily take iterative analyses into account. The model is useful for the delivery of documentation and project milestones, but it may not be well suited to a well-engineered product.

The waterfall model is inherent in the communication of the military acquisition process. For example, the Air Force user writes some type of top-level requirements document that then gets accepted by the Air Force procuring agent. The procuring agent then specifies and performs all contract monitoring for the system until the system is completed. The agent then turns the system over to the user. While it is true that the user and procuring agent often try very hard to work together, there are not enough iterations of design analyses to ensure that the system meets its mission criticality requirements.

A large portion of the mis-specification problem revolves around inadequate guidance. Computer security and software safety are evolving very quickly, and rapidly generate new requirements. For instance, in 1989, DoD directive 5200.28 specified that all government systems must have the equivalent of a C2 COMPUSEC rating by 1992 [17]. A draft requirement for Trusted Critical Computer Systems Evaluation Criteria [4] is being developed which uses data integrity and denial of service as its main tenets. The relationships of these documents to the acquisition cycle has not been sufficiently specified, nor have the appropriate guidance documents been written. However, COMPUSEC requirements do have an advantage over software safety requirements, because the latter are not nearly as well specified.

Air Force system development generally adheres to the concept of requirement flowdown. An executive order [18] appoints the authority to determine policy. DoD 5200.28-D is a top-level computer security policy for application within the Department of Defense. The TCSEC (DoD-5200.28-STD), specifies system requirements. The Technical Rationale Behind [DoD-5200.28-STD] Computer Security Requirements (CSC-STD-004-85) [13] specifies the minimal TCSEC system security classes according to the system environment. Military Standards are usually written by various military commands according to specific type of program. For example, MIL-STD-1785, System Security Engineering Program Management [23], is an Air Force standard on how to manage system security. Different Air Force organizations have their own attachment of Air Force Regulation 205-16, which specifies the accreditation process and some design, application, and operation requirements.

Software safety is a discipline within system safety, and as such adheres to the general requirements established by system safety. However, there are no executive orders, top-level policies, or DoD requirements that establish software safety standards for the Air Force or any other military command. There are only incomplete attempts to establish such standards. For example, MIL-STD-882B describes a system safety acquisition process that in turn describes the software safety process. AFR 800-16 describes Air Force safety programs, while Air Force Consolidated Space Test Center (CSTC) Regulation 127-1 specifies the safety accreditation process to be used by CSTC programs. Handbooks on a range-by-range basis only provide "guidance" for implementing the regulation requirements.

The TCSEC computer security requirements for a class-A system are listed in Table 3, where they are mapped to software development and software safety requirements documents. The mapping indicates if there is a requirement overlap, if there is a requirement dual, or if there is no current overlap. A requirement overlap indicates that there are similar TCSEC and software safety goals and methods. A requirement dual indicates that the TCSEC and software safety implementation details are not the same, but their intent is similar. No overlap indicates that there are no similar requirements or intents.

Table 3. Trusted Computer System Evaluation Criteria to Software Development and Software Safety Requirements Mapping

DISCRETIONARY ACCESS CONTROL	OBJECT REUSE	LABELS	LABEL INTEGRITY	EXPORTATION OF LABELED INFO	EXPRTN OF MULTILEVEL DVCS	EXPRTN TO SNGL-LEVEL DVCS	LABELING HUMAN-READABLE OUTPUT	MANDATORY ACCESS CONTROL	SUBJECT SENSITIVITY LABELS	DEVICE LABELS	IDENTIFICATION & AUTHENTICATION	AUDIT	TRUSTED PATH	SYSTEM ARCHITECTURE	SYSTEM INTEGRITY	SECURITY TESTING	DESIGN SPECIFICATION & VERIFICATION	COVERT CHANNEL ANALYSIS	TRUSTED FACILITY MANAGEMENT	CONFIGURATION MANAGEMENT	TRUSTED RECOVERY	TRUSTED DISTRIBUTION	SECURITY FEATURES USER'S GUIDE	TRUSTED FACILITY MANUAL	TESTED DOCUMENTATION	DESIGN DOCUMENTATION	
																X	X								X	X	MIL-STD-1521B
																X	X								X	X	MIL-STD-490A
														X		X	X								X	X	DoD-STD-2167
																				X							DoD-STD-2168
												▨			▨	▨	X	▨		X	▨				X	X	MIL-STD-882B
																▨			▨	X			▨	▨	X		CSTCR 127-1
SECURITY POLICY											ACCOUNTABILITY			ASSURANCE									DOCUMENTATION				

□ NO REQUIREMENT OVERLAP   X REQUIREMENT OVERLAP   ▨ REQUIREMENT DUAL

All of the TCSEC documentation and assurance requirements categories, except trusted distribution, have analogous software development or software safety requirements; some mappings indicate requirement overlap, while some indicate requirement duals.

The TCSEC design and test documentation requirements are similar to those specified in almost all of Air Force software development and software safety requirements documents; in fact, all have implicit waterfall development cycles [8]. For example, security-testing overlaps software development testing, validation, and verification, as well as the software-safety requirements testing specified in MIL-STD-882B, series 300. Configuration management for the TCSEC and software safety overlap, since both track relevant mechanisms. The design specification and verification required in the TCSEC map to several software design and software safety specifications. With regard to complexity and modularity, the TCSEC system architecture requirement is similar to software development in DoD STD-2167.

The Security Features User's Guide and Trusted Facility Manuals are duals with regard to Operating Instruction requirements for the safety accreditation of deliverables per CSTCR 127-1. Trusted recovery for software safety is required in the sense that a system should have a known point at which processing is resumed in accordance with MIL-STD-882B, but it is not as well defined there as it is in the TCSEC. Covert-channel analysis is a dual of a sneak circuit or soft tree analyses. Audit is similar to error detection and reporting in MIL-STD-882B.

There are no current software safety requirements for implementing any of the security policy or accountability requirements except audit. This does not mean that computer security policy and accountability requirements cannot be used to implement safety or safety-related features (with the exception of policies and assurances based on disclosure), but it may indicate that software safety has not reached the level of requirements sophistication needed to specify features clearly on the basis of criticality or integrity. It may be possible to combine features to provide trust for both computer security and software safety; it may also be possible to implement safety functions using traditional security ideas and mechanisms. This paper suggests that both of these approaches be applied to implement software safety and computer security.

Many of the COMPUSEC and software safety assurance and documentation requirements are meant to ensure proper design. Both utilize computers and software, and rely on good programming practices. Both require knowledgeable computer professionals at all levels to implement the systems properly. If software professionals take security and safety into consideration, it is much easier to communicate concerns across the software, security, and safety disciplines. Ensuring proper development is also the goal of MIL-STDs 2167/2168 and 1521B. The design requirements of trusted systems must be integrated into those military standards so that there is little or no duplication of effort and to ensure that the efforts are coordinated.

Neither computer security nor software safety has been well integrated into the acquisition cycle [25, 10]. Accreditation requirements do not flow into the acquisition phase, and design, integration, and verification are not in synchrony with the DoD-STDs 2167 [8] and 2168, or in MIL-STDs 1521B and 490. The integration of computer security and software safety may be better served by a model that allows for iteration or feedback between phases. The spiral model [9] of software development may capture the repetition whose risk-driven approach provides a framework for guiding the software process.

> This paper does not advocate that computer security or software safety requirements be simply "forced" into DoD-STDs 2167 and 2168 and MIL-STD-1521B. Rather, this paper advocates that these documents be rewritten, in accordance with an iterative model, to provide for design iteration and to permit system designers to reduce safety and security vulnerabilities. A discussion of what is involved in this rewriting effort is beyond the scope of this paper.

ACCREDITATION PROCESS INTEGRATION

The accreditation process and its necessary inputs are very similar for ADPS System Security in AFR 205-16 and Software Safety in CSTC 127-1. ADPS security and software safety each require accreditation by separate organizations. For example, in the ADPS security arena this individual is the Designated Approving Authority (DAA). However, the inputs necessary for accreditation are provided by different individuals, who are responsible for documenting and ensuring both safety and security in day-to-day operations. In the security arena the Computer System Security Officer (CSSO) is responsible for accreditation inputs and maintenance. The evidence the CSSO presents to the DAA for approval includes information about the environment, hardware, software, personnel, operations, mission, and contingency plans, as well as a comparison of all the computer security and software safety requirements against a known system baseline. The risks are identified and assessed, then mitigation techniques are applied. Residual risk is then presented to the DAA. Obviously, this process is confusing and leads to a variety of problems.

Therefore, a single accreditation cycle for both computer security and software safety makes sense: there is only one approval authority and one responsible officer, information gets presented only once, resources are used only once, and the risks are presented as *integrated risks*. Another added benefit of a unified accreditation cycle is that integrated risks are easily represented to the various working groups.

To illustrate this point, consider a secure priority-A satellite control station that must operate without interruption through laborious redundant database and hardware backups and whose contingency procedures have been well honed. This equipment is kept within a locked and guarded room whose access requirements are strict. The safety regulations for the building require an emergency power shutoff outside the module, so that Air Force personnel may access it easily during an emergency. However, to meet security regulations for uninterrupted service, this switch should never be placed where it could be activated inadvertently, unintentionally, or by an adversary. Obviously the requirements for security and safety are in conflict here; the risk represented by the switch was not considered as an *integrated* risk.

Combining the accreditation of security and safety is especially effective when a system is changing rapidly. As an example, consider a large computer program that is used by hundreds of users to control satellites. The program provides tracking, telemetry, and control, and is provided and maintained by a single organization. This program is considered off-the-shelf (OTS) to all new users, since it was procured ten years ago. Now imagine that this system must support forty missions, all of which are critical, each with its own mission-unique equipment (MUE) consisting of a custom operator interface and mission-unique data. To satisfy this need, it is possible to accredit the OTS for computer security and software safety under a proposed integrated Trusted System accreditation, while preserving that accreditation intact until a major system change requires a new one. The MUE, on the other hand, can be accredited as required, with the end user—rather than the service providers—submitting the evidence. The service providers have the authority to deny the use of a specific MUE until it meets approval. Accrediting the MUE independently of the OTS provides the most flexibility.

> This paper proposes that accreditation requirements be specified during the acquisition process.

These requirements must encompass the complete temporal utilization of the computing system: boot/load, hardware integrity checks, system configuration, mission-data loading, mission performance, changes in mission processing due to security level changes or user privileges, contingency plans and reduced modes of operation, task deletion by criticality, reinitialization and restart, recovery, backup and data restoration, shutdown, and maintenance.

COMPUSEC MECHANISMS IMPLEMENTATION OF SOFTWARE SAFETY

The prevention of hazard and threat vulnerabilities requires that the vulnerability path be identified and that a specific mechanism be used to ensure that this path cannot be activated. Two concerns arise when the path-blocking mechanism is designed: will it mitigate the vulnerability under all appropriate conditions, and is the mechanism itself secure from alteration? Computer security and software safety path blocking assurances and documentation are similar. The foundations, control goals, access basis, and access applications are similar.

> Since these similarities exist, this paper proposes that TCSEC policy features be used to implement *trusted* systems that include both computer security and software safety.

The COMPUSEC concept of a Trusted Computing Base (TCB) is that all protection mechanisms used to enforce the *security policy* for the system are contained within this trusted base. The TCB is as small as possible to permit verification, is modular, has well-defined interfaces, and contains only that code essential to the security policy. However, the TCB concept can also be applied to software safety, to enforce *safety policy*. In either case, the reference monitor in the TCB must have three principle properties: it must be uncircumventable, it must be tamper-proof, and it must be always invoked. Nonetheless, the issue of including safety mechanisms within the TCB is controversial, since this may make the TCB larger and more complex.

Therefore, early in the requirements definition phase, safety-relevant elements must be identified and classified, in order to identify common mechanisms and safety-relevant databases for inclusion in the TCB. The databases must be protected by TCB security mechanisms and must allow access to only those users with proper privileges.

Important to both computer security and software safety is the concept of *data integrity*. In computer security, object reuse prevents disclosure of data through scavenging. Object reuse prevention is the clearing of user-addressable memory before that memory is allocated for subsequent use. Object reuse prevention designs include clearing upon memory deallocation or clearing before memory allocation. The former approach is an important feature for software safety, since it is important that all unused memory locations (if executed) be benign. Since object reuse sets all bits to zero, it is important that the software safety design take into account the interpretation of all command words whose contents are zeros; an example of this is a benign command such as a no-operation (no-op).

Safety features such as interlocks and sequence checks can be contained in safety modules within the TCB. Identification and authentication mechanisms can provide safing or override safing. The audit mechanisms can be used to detect hazards and produce alarms. In a failover situation, Discretionary Access Control (DAC) mechanisms can be used to grant a redundant processor access to data from the failed processor; in the event of the failure of the primary control processor in an N-version programmed system, DAC mechanisms can grant system control.

A simple example of building a TCB based on security and safety follows. Consider a weapons system in which an operator has the capability to override failures that could lead to disclosure or a hazard when the weapon system is delivered. In this case a TCB containing a fail-operational system is appropriate. The weapons system could require an additional privilege for both normal weapons delivery and weapons delivery upon failure in the fail-operational system. Once either of the privileged modes was entered, a privileged trusted process would activate the fail-operational design and would not permit control to be altered by any other process. That privileged trusted process would then resume normal operation only if the manual control were overridden or the fault were cleared. Entry into either of the privileged modes could require an additional Identification and Authentication (I&A). This type of mission-critical override could also require a different type of I&A than that needed to access the system. For instance, the traditional implementation of I&A is the login and password. For a quick I&A in a hostile environment, it may be better to use anthropomorphic mechanisms (e.g., retinal scans, fingerprints, voice recognition).

As a generic example, the hazard prevention method of lockout could be implemented inside the TCB. This mechanism should be invoked or revoked only by a process with the authority to do so. This process must be a trusted process, i.e., one that has been verified as having the correct level of assurance that it will work as required and is contained within the TCB. In order to request the invocation or revocation of the lockout, this trusted process must ensure that the correct labels are on the information. To enforce the correct processing of these labels and produce the lockout, the labels must relay on a Mandatory Access Control (MAC) mechanism within the TCB.

CONCLUSION

Software safety and COMPUSEC have many overlapping goals, requirements, mitigating features, assurances, and both acquisition and accreditation processes. This paper suggests that these two subdisciplines within system and software development be integrated for mutual benefit as *trusted* system development. This paper suggests the use of some of the traditional COMPUSEC mechanisms for implementing software safety. An integrated approach to mitigating vulnerabilities and combing common requirements, analyses, evidence of assurance, and milestones will lead to a better product.

The following is a summary of our recommendations:

- Traditional computer security mechanisms must be studied for use as software-safety design features.

- Both system and software engineers must be trained to design and implement integrated computer security and software safety as *trusted* computing.

- The requirements must be rewritten for trusted critical systems. Instead of the currently separate computer security and software safety requirements, the new requirements must encompass criticality, computer security, and software safety. Table 4 maps the document levels required for *trusted* system documentation to examples of current computer security or software safety documents.

- The acquisition and accreditation processes must be better integrated to provide an iterative process of development, rather than the traditional waterfall model.

- Finally, other procurement requirements documents, such as MIL-STD-1521B, DoD-STD-2167, and DoD-STD-2168, must be updated to support the proposed approach to trusted-system development.

Table 4. Trusted System Requirement Document Levels.

Document Level	Current Documents
Executive Order	EO 12356
DoD Directive	DoD 5200.28-D
DoD Application	CSC-STD-004-85
DoD Standard	DoD 5200.28-STD
MIL-STD	MIL-STD-882B, MIL-STD-1785
Air Force Reg.	AFR205-16, CSTCR 127-1

REFERENCES

[1] AFISC SSH 1-1. *Software System Safety, Headquarters Air Force Inspection Safety Center, September 5, 1985.*

[2] AFR205-16. *Department of the Air Force Automatic Data Processing (ADP) Security Policy, Procedures, and Responsibilities.*

[3] Air Force HQ Electronic Security Command. *Computer Critical Mission Security Requirements and Technical Rationale, 31 July 1989.*

[4] Air Force HQ Electronic Security Command. *Trusted Critical Computer System Evaluation Criteria (TSSEC) (Draft), 31 July 1989.*

[5] Air Force HQ Electronic Security Command. *Trusted System Evaluation Criteria Interpretation for Embedded Computer Systems, 26 July 1989.*

[6] Air Force Space Test Center. *Regulation 127-1 (CSTCR 127-1), Space Test Range Safety.*

[7] Air Force Space Test Center. *CSTC Handbook 127-1, Space Safety User's Handbook.*

[8] Benzel, T. C. Vickers. *"Integrating Security Requirements and Software Development Standards,"* Proceedings of the 12th National Computer Security Center Conference, *Baltimore, Maryland, October 1989.*

[9] Boehm, B. W., *"A Spiral Model of Software Development and Enhancement," IEEE Computer, May 1988.*

[10] Brown, M. L. *"Tailoring MIL-STD-882B 300 Series Tasks,"* Proceedings of the Ninth International System Safety Conference, *July 1989.*

[11] Clark, D. D., Wilson, D. R., *"Evolution of a Model for Computer Integrity,"* Proceedings of the 11th National Computer Security Conference, *Baltimore, Maryland, October 1988.*

[12] Crocker, S. D., *"Techniques for Assuring Safety—Lessons from Computer Security,"* Proceedings of the Computer Assurance Conference, *Washington, DC, July 1987.*

[13] CSC-STD-004-85. *Technical Rationale Behind CSC-STD-003-85.*

[14] DoD-STD-2167A. *Defense System Software Development.*

[15] DoD-STD-2168. *Defense System Software Quality Program.*

[16] DoD 5200.28-STD. *Department of Defense Trusted Computer System Evaluation Criteria (the Orange Book).*

[17] DoD Directive 5200.28. *Security Requirements for Automatic Data Processing (ADP) Systems.*

[18] Executive Order 12356. *National Security Information.*

[19] Lavine, C., Thomas, Hoheb, Baker. *"Survey of Software Safety Programs,"* Proceedings of the Ninth International System Safety Conference, *July 1989.*

[20] MIL-STD-490A. *Engineering Management.*

[21] MIL-STD-882B Notice 1. *System Safety Program Requirements (30 March), U.S. Department of Defense, U.S. Government Printing Office, Washington D.C., 1984.*

[22] MIL-STD-1521B. *Technical Reviews and Audits for Systems, Equipment, and Computer Software.*

[23] MIL-STD-1785. *System Security Engineering Program Management Requirements.*

[24] Neumann, P. G., Parker. *"A Summary of Computer Misuse Techniques,"* Proceedings of the 12th National Computer Security Center Conference, *Baltimore, Maryland, October 1989.*

[25] Norvell, W., *"Integration of Security into the Acquisition Life Cycle,"* Proceedings of the National Security Industrial Association, *San Diego, CA, 18–20 October 1988.*

[26] Spafford, E. H., *"The Internet Worm Program: An Analysis," ACM SIGCOM, vol. 19, January 1989.*

[27] Thomas, J. C., *"The Use of Kernel Architectures for Software Safety," The Aerospace Corporation, El Segundo, CA.*

INTEGRITY MECHANISMS IN DATABASE MANAGEMENT SYSTEMS

Ravi Sandhu and Sushil Jajodia

Department of Information Systems and Systems Engineering
George Mason University, Fairfax, VA 22030-4444

Abstract. Our goal in this paper is to answer the following question: what mechanisms are required in a general-purpose multiuser database management system (DBMS) to facilitate the integrity objectives of information systems? We are particularly interested in relational DBMS's. Although existing commercial products fall far short of providing the requisite mechanisms, in principle they can be easily extended to incorporate these mechanisms. In a nutshell our conclusion is that realistic mechanisms do exist. Our principal contribution is to identify these mechanisms, fill in the gaps where none existed and point out where gaps still remain. We have also bridged the terminology and concepts of database and security specialists in a coherent manner.

1 INTRODUCTION

Information integrity means different things to different people, and will probably continue to do so for some time. The recent NIST workshop, which set out to establish a consensus definition, instead arrived at the following conclusion [18, page 2.6].

> The most important conclusion to be drawn from this compilation of papers and working group reports: don't draw too many conclusions about the appropriate definition for data integrity just yet. ... In the mean time, papers addressing integrity issues should present or reference a definition of integrity applicable to that paper.

So the first order of business is to define integrity. Our approach to this question is pragmatic and utilitarian. The objective is to settle on a definition within which we can achieve practically useful results, rather than searching for some absolute and airtight formulation.

We define integrity[1] as being concerned with the *improper modification* of information (much as confidentiality is concerned with improper disclosure). We understand modification to include insertion of new information, deletion of existing information as well as changes to existing information.

The reader has probably seen similar definitions using "unauthorized" instead of "improper." Our use of the latter term is quite deliberate and significant. Firstly, it acknowledges that security breaches can and do occur without authorization violations, i.e., authorization is only one piece of the solution. Secondly, it adheres to the well-established and useful notion that information security has three components: integrity, confidentiality and availability. We see no need to discard this standard viewpoint in the absence of some compelling demonstration of a superior one. Finally, our definition brings to the front the very important question: what do we mean by improper? It is obvious that this question intrinsically cannot have an universal answer. So it is futile to try to answer it outside of a given context.

[1] We should point out that our definition of integrity is considerably broader than the traditional use of this term in the database literature. For instance Date [6] says: "Security refers to the protection of data against unauthorized disclosure, alteration, or destruction; integrity refers to the accuracy or validity of data." The consensus view among security researchers is that integrity is one component of security and accuracy/validity is one component of integrity [9, 18, for instance].

We are specifically interested in information systems used to control and account for an organization's assets. In such systems the primary goal is prevention of fraud and errors. The meaning of improper modification in this context has been given by Clark and Wilson [2] as follows.

> No user of the system, even if authorized, may be permitted to modify data items in such a way that assets or accounting records of the company are lost or corrupted.

Note their express qualification: "even if authorized." The word company in this quote reveals the authors' commercial bias but, as they have clarified [3], these concepts apply equally well to any information system which controls assets—be it in the military, government or commercial sectors.

Our goal in this paper is to answer the following question: what mechanisms are required in a general-purpose multiuser DBMS to help achieve the integrity objectives of information systems? There are many compelling reasons to focus on DBMS's for this purpose. The most important one has been succintly stated by Burns [1] as follows.

> A database management system (DBMS) provides the appropriate level of abstraction for the implementation of integrity controls as presented in the Clark and Wilson paper [2]. ... It is clear that the domain of applicability of the Clark and Wilson model is not an operating system or a network or even an application system, it is fundamentally a DBMS.

This is particularly true when we focus on mechanisms. Moreover DBMS's have the wonderful ability to express and manipulate complex relationships. This comes in very handy when dealing with sophisticated integrity policies.

The Operating System (OS) must clearly provide some core integrity and security mechanisms. In terms of the Orange Book [8] one would need at least a B1 system to enforce encapsulation of the DBMS, i.e., to ensure that all manipulation of the database can only be through the DBMS. The question of what minimal features are required in the OS is an important one but outside the scope of the present paper. For now let us assume that OS's with the requisite features are available.

The bulk of integrity mechanisms belong in the DBMS. Integrity policies are intrinsically application specific and the OS simply cannot provide the means to state application specific concerns. One might then argue: why not put all the mechanism in the application code? There are several persuasive reasons not to do this. Firstly, it is not very conducive to reuse of common mechanisms. Secondly, any assurance that mechanisms interspersed within application code will be correct or even comprehensible is rather dubious. Thirdly, the whole point of a database is to support multiple applications. A particular application may well be in a position to handle all its integrity requirements. Yet it is only the DBMS which can prevent other applications from courrupting the database.

The rest of the paper is organized as follows. In section 2 we discuss principles for achieving integrity in information systems. In section 3 we describe the mechanisms required in a DBMS to support these high level principles. In some of the more detailed consideration we will limit ourselves specifically to relational DBMS's. As we will see traditional DBMS mechanisms provide the foundations for this purpose, but by themselves do not go far enough. Section 4 concludes the paper.

2 INTEGRITY PRINCIPLES

We begin by describing basic principles for achieving information integrity. These principles can be viewed as high level objectives which are made more concrete when specific mechanisms are proposed to support them. In other words these principles lay down broad goals without specifying how to achieve them. We will subsequently map these principles to DBMS mechanisms. We emphasize that the principles themselves are independent of the DBMS context. They apply equally well to any information system be it a manual paper-based system, a centralized batch system, an interactive and highly distributed system, etc.

The nine integrity principles enumerated below are abstracted from the Clark and Wilson papers [2, 3, 4], the NIST workshops [17, 18] and the broader security and database literature.[2] The reader has probably seen similar lists in the past. We believe the numerous discussions spurred by the Clark-Wilson papers call for a revised formulation of major principles. We emphasize that these principles express *what* needs to be done rather than *how* it is going to be accomplished. The latter question is addressed in the next section.

1. *Well-formed Transactions.* Clark and Wilson [2] have defined this principle as follows: "The concept of the well-formed transaction is that a user should not manipulate data arbitrarily, but only in constrained ways that preserve or ensure the integrity of the data." This principle has also been called *constrained change* [4], i.e., data can only be modified by well-formed transactions rather than by arbitrary procedures. Moreover the well-formed transactions are known ("certified") to be individually correct with some (mostly qualitative) degree of assurance.

2. *Authenticated Users.* This principle stipulates that modifications should only be carried out by users whose identity has been authenticated to be appropriate for the task.

3. *Least Privilege.* The notion of least privilege was one of the earliest principles to emerge in security research. It has classically been stated in terms of processes (executing programs) [19], i.e., a process should have exactly those privileges needed to accomplish its assigned task, and none extra. The principle applies equally well to users, except that it is more difficult to precisely delimit the scope of a user's "task." A process is typically created to accomplish some very specific task and terminates on completion. A user on the other hand is a relatively long-lived entity and will be involved in varied activities during his lifespan. His authorized privileges will therefore exceed those strictly required at any given instant. In the realm of confidentiality least privilege is often called *need-to-know.* In the integrity context it is appropriately called *need-to-do.* Another appropriate term for this principle is *least temptation*, i.e., do not tempt people to commit fraud by giving them greater power than they need.

4. *Separation of Duties.* Separation of duties is a time honored principle for prevention of fraud and errors, going back to the very beginning of commerce. Simply stated, no single individual should be in a position to misappropriate assets on his own. Operationally this means that a chain of events which affects the balance of assets must require different individuals to be involved at key points, so that without their collusion the overall chain cannot take effect.

5. *Reconstruction of Events.* This principle seeks to deter improper behavior by threatening its discovery. It is a necessary adjunct to least privilege for two reasons. Firstly least privilege, even taken to its theoretical limit, will leave some scope for fraud. Secondly a zealous application of least privilege is not a terribly efficient way to run an organization. It conveys an impression of an enterprise enmeshed in red tape.[3] So practically users must be granted more privileges than are strictly required. We therefore should be able to accurately reconstruct essential elements of a system's history, so as to detect misuse of privileges.

6. *Delegation of Authority.* This principle fills in a piece missing from the Clark and Wilson

[2] The literature is too numerous to cite individually. For those unfamiliar with the "older" literature some useful staring points are [7, 9, 10, 13, 19].

[3] This comment is made in the context of users rather than processes (transactions). Least privilege with respect to processes is more of an internal issue within the computer system, and its zealous application is most desirable (modulo the performance and cost penalties it imposes).

papers and much of the discussion they have generated.[4] It concerns the critical question of how privileges are acquired and distributed in an organization? Clearly the procedures to do so must reflect the structure of the organization and allow for effective devolution of authority. Individual managers should have maximum flexibility regarding information resources within their domain, tempered by constraints imposed by their superiors. Without this flexibility at the end-user level, the authorization will most likely be inappropriate to the actual needs. This can only result in security being perceived as a drag on productivity and something to be bypassed whenever possible.

7. *Reality Checks.* This principle has been well motivated by Clark and Wilson [4] as follows: "A cross-check with the external reality is a central part of integrity control. ... integrity is meaningful only in terms of the relation of the data to the external world." Or in more concrete terms: "If an internal inventory record does not correctly reflect the number of items in stock, it makes little difference if the value of the recorded inventory has been reflected correctly in the company balance sheet."

8. *Continuity of Operation.* This principle states that system operations should be maintained to some appropriate degree in the face of potentially devastating events which are beyond the organization's control. This catch-all description is intended to include natural disasters, power outages, disk crashes and the like.[5]

9. *Ease of Safe Use.*[6] In a nutshell this principle requires that the easiest way to operate a system should also be the safest. There is ample evidence that security measures are all too often incorrectly applied or simply bypassed by the system managers. This happens due to a combination of (i) poorly designed defaults (such as indefinite retention of vendor-supplied passwords for privileged accounts), (ii) awkward and cumbersome interfaces (such as requiring many keystrokes to effect simple changes in authorization), (iii) lack of tools for authorization review, or (iv) mismatched policy and mechanism ("...the extent that the user's mental image of his protection goals matches the mechanism he must use, mistakes will be minimized." [19]).

It is inevitable that these principles are fuzzy, abstract and high level. In developing an organization's security policy one would elaborate on each of these principles and make precise the meaning of terms such as "appropriate" and "proper." How to do so systematically is perhaps the most important question in successful application of these principles. In other words how does one articulate a comprehensive policy based on these high level objectives? This question is beyond the scope of this document. Our present focus is on the question: how do these principles translate into concrete mechanisms in a DBMS?

The goals encompassed by these principles may appear overwhelming. After all in the extreme these principles amount to solving the total system correctness problem, which we know is well beyond the state of the art. Fortunately, in our context, the degree to which one would seek to enforce these objectives and the assurance of this enforcement are matters of risk management and cost-benefit analysis. Laying out these principles explicitly does give us the following major benefits.

- The overall problem is partitioned into smaller components for which solutions can be developed independently of each other (i.e., divide and conquer).
- The principles suggest common mechanisms which belong in the DBMS and can be reused across multiple applications.

[4] The closest concept that Clark and Wilson have to this principle is their Rule E4 which they summarize as follows [2, figure 1]: "Authorization lists changed only by the security officer." This notion of a central security officer as an authorization czar is inappropriate and unworkable. Rational security policies can be put in place only if appropriate authority is vested in end-users.

[5] One might argue that we are stepping into the scope of availability here. If so, so be it.

[6] Thanks to Stanley Kurzban and William Murray for coining this particular term.

- The principles provide a set against which the mechanisms of specific DBMS's can be evaluated (in an informal sense).
- The principles similarly provide a set on the basis of which the requirements of specific information systems can be articulated.
- Last, but not the least, the principles invite criticism from the security community particularly regarding what may have been left out.

3 INTEGRITY MECHANISMS

In this section we consider DBMS mechanisms to facilitate application of the principles defined in the previous section. The principles have been applied in practise [15, 26, for instance] but with most of the mechanism built into application code. Providing these mechanisms in the DBMS is an essential prerequisite for their widespread use.

Our mapping of principles to mechanisms is summarized in table 1. Some of these mechanisms are available in commercial products. Others are well established in the database literature. There are also some newer mechanisms which have been proposed more recently, e.g., transaction controls for separation of duties [21], the temporal model for audit data [12] and propagation constraints for dynamic authorization [20, 22]. Finally there are places where existing mechanisms and proposals need to be extended in novel ways. Overall the required mechanisms are quite practical and well within the reach of today's technology.

3.1 Well-formed Transactions

The concept of a well-formed transaction corresponds very well to the standard DBMS concept of a transaction [10, 11]. A transaction is defined as a sequence of primitive actions which satisfies the following properties.

1. *Failure atomicity*: either all or none of the updates of a transaction take effect. We understand update to mean modification, i.e., it includes insertion of new data, deletion of existing data and changes to existing data.
2. *Serializability*: the net effect of executing a set of transactions is equivalent to executing them in some sequential order, even though they may actually be executed concurrently (i.e., their actions are interleaved or simultaneous).
3. *Progress*: every transaction will eventually complete, i.e., there is no indefinite blocking due to deadlock and no indefinite restarts due to livelocks.
4. *Correct state transform*: each transaction if run by itself in isolation and given a consistent state to begin with will leave the database in a consistent state.

We will elaborate on these properties in a moment.

First let us note the basic requirement that the DBMS must ensure that updates are restricted to transactions. Clearly, if users are allowed to bypass transactions and directly manipulate relations in a database, we have no foundation to build upon. We represent this requirement by the diagram in figure 1. In other words updates are encapsulated within transactions. At this point it is worth recalling that the database itself must be encapsulated within the DBMS by the Operating System.

It is clear that the set of database transactions is itself going to change during the system life cycle. Now the same nine principles of the previous section apply with respect to maintaining the

INTEGRITY PRINCIPLE	DBMS MECHANISMS
Well-formed transactions	Encapsulated updates Atomic transactions Consistency constraints
Authenticated users	Authentication
Least privilege	Fine grain access control
Separation of duties	Transaction controls Layered updates
Reconstruction of events	Audit trail
Delegation of authority	Dynamic authorization Propagation constraints
Reality checks	Consistent snapshots
Continuity of operation	Redundancy Recovery
Ease of safe use	Fail-safe defaults Human factors

Table 1: Summary

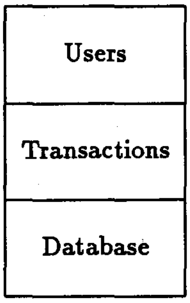


Figure 1: Encapsulated Updates

integrity of the transactions. In particular transactions should be installed, modified and supplanted only by the use of well-formed "transaction-maintenance transactions." One can apply this argument once again to say that the transaction-maintenance transactions themselves need to be maintained by another set of transactions, and so on indefinitely. We believe there is little to be gained by having more than two steps in this potentially unbounded sequence of transaction-maintenance transactions. The rate of change in the transaction set will be significantly slower than the rate of change in the data base proper. Going one step further, the rate of change in the transaction-maintenance transactions will be yet slower to the point where, for all practical purposes, these can be viewed as static over the lifespan of typical systems. With this perspective the data base administrator is responsible for installing and maintaining transaction-maintenance transactions, which in turn control the maintenance of actual database transactions.

We now return to considering the four properties of DBMS transactions enumerated earlier. The first three properties—failure atomicity, serializability and progress—can be achieved in a purely "syntactic" manner, i.e., completely independent of the application. These three requirements for a transaction are recognized in the database literature as appropriate for the DBMS to implement. Mechanisms to achieve these objectives have been extensively researched in the last fifteen years or so, and our understanding of this area can certainly be described as mature. The basic mechanisms—two-phase locking, timestamps, multi-version databases, two-phase commit, undo-redo logs, shadow pages, deadlock detection and prevention—have been known for a long time and have made their way into numerous products. In developing integrity guidelines and/or evaluation criteria one might consider some progressive measure of the extent to which a particular DBMS meets these objectives. For instance, with failure atomicity, is there a guarantee that we will know which of the two possibilities occurred? Similarly, with serializability, does the DBMS enforce the concurrency control protocol or does it rely on transactions to execute explicit commands for this purpose? And, with the issue of progress, do we have a probabilistic or absolute guarantee? Such questions must be systematically addressed.

The fourth property of correct state transforms is the ultimate bottleneck in realizing well-formed transactions. It is also an objective which cannot be achieved without considering the semantics of the application. The correctness issue is of course undecidable in general. In practice we can only assure correctness to some limited degree of confidence by a mix of software engineering techniques such as formal verification, testing, quality assurance, etc. Responsibility for implementing transactions as correct state transforms has traditionally been assigned to the application programmer. Even in theory DBMS mechanisms can never fully take over this responsibility.

DBMS mechanisms can help in assuring the correctness of a state by enforcing *consistency constraints* on the data. Consistency constraints are also often called integrity constraints or integrity rules in the database literature. Since we are using integrity in a wider sense we prefer the former term.

The relational data model in particular imposes two consistency constraints [5, 6].

- *Entity integrity* stipulates that attributes in the primary key of a base relation cannot have null values. This amounts to requiring that each entity represented in the database must be uniquely identifiable.
- *Referential integrity* is concerned with references from one entity to another. A foreign key is a set of attributes in one relation whose values are required to match those of the primary key of some specific relation. Referential integrity requires that a foreign key either be all null[7] or a matching tuple exist in the latter relation. This amounts to ruling out dangling references to non-existent entities.

Entity integrity is easily enforced. Referential integrity on the other hand requires more effort and has seen limited support in commercial products. The precise manner in which to achieve it is also very dependent on the semantics of the application. This is particularly so when the referenced tuple is deleted. There are several choices as follows: (i) prohibit this delete operation, (ii) delete the referencing tuple (with a possibility of further cascading deletes), or (iii) set the foreign key attributes in the referencing tuple to null. There are proposals for extending SQL so that these choices can be specified for each foreign key.

The relational model in addition encourages the use of *domain constraints* whereby the values in a particular attribute (column) are constrained to come from some given set. These constraints are particularly easy to state and enforce, at least so long as the domains are defined in terms of primitive types such as integers, decimal numbers and character strings. A variety of *dependency constraints* which constrain the tuples in a given relation have been extensively studied in the database literature.

In the limit a consistency constraint can be viewed as an arbitrary predicate that all correct states of the database must satisfy. The predicate may involve any number of relations. Although this concept is theoretically appealing and flexible in its expressive power, in practice the overhead in checking the predicates for every transaction has been prohibitive. As a result relational DBMS's typically confine their enforcement of consistency constraints to domain constraints and entity integrity.

3.2 Continuity of Operation

The problem of maintaining continuity of operation in the face of natural disasters, hardware failures and other disruptive events has received considerable attention in both theory and practice [10]. The basic technique to deal with such situations is redundancy in various forms. Recovery mechanisms in DBMS's must also ensure that we arrive at a consistent state. In many respects these mechanisms are "syntactic" in the sense of being application independent, much as mechanisms for the first three properties of section 3.1 were.

3.3 Authenticated Users

Authentication is primarily the responsibility of the Operating System. If the Operating System is lacking in its authentication mechanism it would be very difficult to ensure the integrity of the DBMS itself. The integrity of the database would thereby be that much more suspect. It therefore makes sense to not duplicate authentication mechanisms in the DBMS.

[7] Often the notion of a null foreign key is semantically incorrect. In such cases an additional consistency constraint can disallow null values.

Authentication underlies some of the other principles, particularly, least privilege, separation of duties, reconstruction of events and delegation of authority. In all of these the end objective can be achieved to the fullest extent only if authentication is possible at the level of individual users.

3.4 Least Privilege

The principle of least privilege translates into a requirement for fine grained access control. We have earlier noted that least privilege must be tempered with practicality in avoiding excessive red tape. Nevertheless a high-end DBMS should provide for access control at very fine granularity, leaving it to the database designers to apply these controls as they see fit.

It is clear from the Clark-Wilson papers, if not evident from earlier work, that modification of data must be controlled in terms of transactions rather than blanket permission to write. We have already put forth the concept of encapsulated updates for this purpose. In terms of the relational model it is not immediately obvious at what granularity of data this should be enforced.

For purpose of controlling read access DBMSs have employed mechanisms based on views (as in System R) or query modification (as in INGRES). These mechanisms are extremely flexible and can be as fine grained as desired. However neither one of these mechanisms provides the same flexibility for flexible control of updates. The fundamental reason for this is our theoretical inability to translate updates on views unambiguously into updates of base relations. As a result authorization to control updates is often less sophisticated than authorization for read access.

In relational systems it is natural for obvious reasons to represent the access matrix by one or more relations [24]. At a coarse level we might control access by tuples of the following form

user, transaction, relation

meaning that the specified user can execute the specified transaction on the specified relation. Tuples of the form shown below would give greater selectivity

user, transaction, relation, attribute

This would allow us to control the execution of transactions such as, "give everyone a 5% raise," without giving the same transaction permission to change employee addresses. The following authorization tuple accomplishes this.

Joe, Give-5%-raise, Employees, Salary

A transaction which gives a raise to a specific employee needs a further dimension of authorization to specify which employee it pertains to. Thus, if Joe is authorized to give a 5% raise to John the authorization tuple would look as follows.

Joe, Give-5%-raise, John, Employees, Salary

We are assuming here that John uniquely identifies the employee receiving the raise. The update is restricted to the Salary attribute of a specific tuple with key equal to 'John' in the Employees relation. So it takes a key, relation and attribute to specify the actual parameter of such a transaction.

Now consider a transaction which moves money from account A to account B, i.e., there are two actual parameters of the transaction. In terms of least privilege we need the ability to bind this transaction to updating the two specific accounts A and B. More generally we will have transactions with N parameters identified in a actual parameter list. So we need authorization tuples of the following form,

user, transaction, actual parameter list

where each parameter in the actual parameter list specifies the item authorized for update by specifying one of the following identifiers

- relation,
- relation, attribute,
- key, relation, attribute.

These three cases respectively give us relation level, "column" level and element level granularity of update control.

It is also important to realize that element-level update authorizations should properly be treated as consumable items. For example, once money has been moved from account A to account B the user should not be able to move it again, without fresh authorization to do so.

3.5 Separation of Duties

Separation of duties finds little support in existing products. Although it is possible to use existing mechanisms for this purpose, these mechanisms have not been designed with this end in mind. As a result their use is awkward at best. This fact was noted by the DBMS group at the 1989 NIST data integrity workshop who concluded their report with the following recommendation [18, section 4.3].

> While the group was able to use existing DBMS features to implement separation of roles controls, we were, however, unable to use existing features in a way that would support easy maintenance and certification. We recommend that data definition and/or consistency check features be enhanced to provide operators that lend themselves to the expression of integrity controls and to allow separation of integrity controls and traditional data.

Separation of duties is inherently concerned with sequences of transactions, rather than individual transactions in isolation. For example consider a situation in which payment in the form of a check is prepared and issued by the following sequence of events.

1. A clerk prepares a voucher and assigns an account.
2. The voucher and account are approved by a supervisor.
3. The check is issued by a clerk who must be different from the clerk in step 1. Issuing the check also debits the assigned account. (Strictly speaking we should debit one account and credit another in equal amounts. The important point for our purpose is that issuing a check modifies account balances.)

This sequence embodies separation of duties since the three steps must be executed by different people. The policy moreover has a dynamic flavor in that a particular clerk can prepare vouchers as well as, on different occasions, issue checks. However he cannot issue a ckech for a voucher prepared by himself.

Implementation of this policy in a paper-based system follows quite directly from its statement.

- The voucher is realized as a form with blank entries for the amount and account, as well as for signatures of the people involved. As the above sequence gets executed these blanks are filled in. On its completion copies of the voucher are filed in various archives for audit purposes.

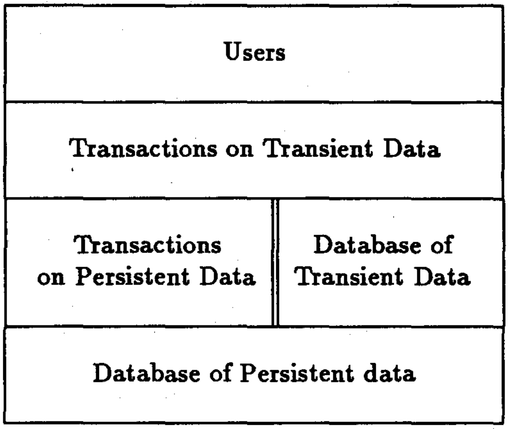

Users
Transactions on Transient Data
Transactions on Persistent Data
Database of Transient Data
Database of Persistent data

Figure 2: Layered Updates

- The account is represented by say a ledger card, where debit and credit entries are posted along with references to the forms which authorized these entries.

By their very nature paper-based controls rely on employee vigilance and internal/external audits for their effectiveness. Computerization brings with it the scope for enforcing the required controls by means of an infallible, ever vigilant and omniscient automaton, viz., the computer itself.

The crucial question is how do we specify and implement similar controls for separation of duties in a computerized environment? A mechanism for this purpose is described in [21]. This mechanism of *transaction-control expressions* is based on the following difference between vouchers and accounts.

- The voucher is *transient* in that it comes into existence, has a relatively small sequence of steps applied to it and then disappears from the system (possibly leaving a record in some archive). The history of a voucher can be prescribed as a finite sequence of steps with an a priori maximum length.

- The account on the other hand is *persistent* in the sense it has a long-lived, and essentially unbounded, existence in the system. During its life there may be a very large number of credit and debit entries for it. Of course, at some point the account may be closed and archived. The key point is that we can only prescribe its history as a variable-length sequence of steps with no a priori maximum length.

Both kinds of objects are essential to the logic and correct operation of an information system. Transient objects embody a logically complete history of transactions corresponding to a unit of service provided to the external world by the organization. Persistent objects embody the internal records required to keep the organization functioning with an accurate correspondence to its interactions with the external world.

Separation of duties is achieved by enforcing controls on transient objects, for the most part. The crucial idea, which makes this possible, is that transactions can be executed on persistent objects only as a side effect of executing transactions on transient objects. This thesis is actually simply borrowed from the paper-based world where it has been routinely applied ever since bookkeeping became an integral part of business operations.

With this perspective we arrive at the diagram shown in figure 2. The idea is that a sequence of transactions is viewed as transient data in the database. In this picture there is a double encapsulation of the database, first by transactions on persistent data and then by transactions on transient data. Users can directly only execute the latter. The former are triggered indirectly as a result when the transient is in the proper state for doing so. In other words transient data is singly encapsulated and has direct application of separation of duties. Persistent data is doubly encapsulated and has indirect application of separation of duties by means of transient data.

3.6 Reconstruction of Events

The ability to reconstruct events in a system serves as a deterrent to improper behavior. In the DBMS context the mechanism to record the history of system is traditionally called an audit trail. As with the principle of least privilege, a high-end DBMS should be capable of reconstructing events to the finest detail. It should also structure the audit trail logically so that it is easy to query. For instance, logging every keystroke does give us the ability to reconstruct the system history accurately. However with this primitive logical structure one needs substantial effort to reconstruct a particular transaction. In addition to the actual recording of all events that take place in the database, an audit trail must also provide support for auditing, i.e., an audit trail must have the capability "for an authorized and competent agent to access and evaluate accountability information by a secure means, within a reasonable amount of time and without undue difficulty" [8]. In this respect DBMSs have a significant advantage, since their powerful querying abilities can be used.

The ability to reconstruct events has different meaning to different people. At one end of the spectrum, we have the requirements of Clark and Wilson [4]. They require only two things:

1. A complete history of each and every modification made to the value of an item.
2. With each change in value of an item, store the identity of the person making the change.

Of course, the system must be reliable in that it makes exactly those changes that are requested by users and the binding of a value with its author is also exact. Clark and Wilson call this "attribution of change."

This can be easily accomplished if we are willing to extend slightly the standard logging techniques for recovery purposes. For each transaction, a recovery log contains the transaction identifier, some *before-images*, and the corresponding *after-images*. If we augment this by recording in addition the user for each transaction, we have the desired binding of each value to its author. There is one other change that needs to be made. In order to support recovery, there is a need to keep a log only up to a point from which a complete database backup is available. Of course, now there is a need to archive the logs so they remain available.

Others have argued that this simple "attribution of change" is not sufficient. We need an audit trail, a mechanism for a complete reconstruction of every action taken against the database: *who* has been accessing *what* data, *when*, and in what *order*. Thus, it has three basic objects of interest:

1. The user - who initiated a transaction, from what terminal, when, etc.
2. The transaction - what was the exact transaction that was initiated.
3. The data - what was the result of the transaction, what were the database states before and after the transaction initiation.

For this purpose a *database activity model* has been recently proposed [12] that imposes a uniform logical structure upon the past, present, and future data. There is never any loss of historical or current information in this model, thus the model provides a mechanism for complete reconstruction of every action taken on the database. It also logically structures the audit data to facilitate its querying.

3.7 Delegation of Authority

The need to delegate authority and responsibility within an organization is essential to its smooth functioning. It appears in its most developed form with respect to monetary budgets. However the concept applies equally well to the control of other assets and resources of the organization.

In most organizations the ability to grant authorization is never completely unconstrained. For example, a department manger may be able to delegate substantial authority over departmental resources to project managers within his department and yet be prohibited to delegate this authority to project managers outside the department. These situations cloud the classic distinction between discretionary and mandatory policies [16, 23]. The traditional concept of ownership as the basis for delegating authority also becomes less applicable in this context [14]. Finally we need the ability to delegate privileges without having the ability to exercise these privileges. Some mechanisms for this purpose have been recently proposed [14, 22].

The complexity introduced by dynamic authorization has been recognized ever since researchers considered this problem, e.g., as stated in the following quote [19].

> "...it is relatively easy to envison (and design) systems that statically express a particular protection intent. But the need to change access authorizations dynamically ...introduces much complexity into protection systems."

This fact continues to be true in spite of substantial theoretical advances in the interim [20]. Existing products provide few facilities in this respect and their mechanisms tend to have an ad hoc flavor.

3.8 Reality Checks

This principle inherently requires activity outside of the DBMS. The DBMS does have obligation to provide an internally consistent view of that portion of the database which is being externally verified. This is particularly so if the external inspection is conducted on an ad hoc on-demand basis.

3.9 Ease of Safe Use

Ease of safe use is more an evaluation of the DBMS mechanisms than something to be enforced by the mechanisms themselves. The mechanisms should of course have fail-safe defaults [19], e.g., access is not available unless explicitly granted or this default rule is explicitly changed to grant it automatically. DBMS's do offer a significant advantage in providing user friendly interfaces intrinsically for their main objective of data manipulation. These interface mechanisms can be leveraged to make the authorization mechanisms easy to use. For instance, having the power of SQL queries to review the current authorizations is a tangible benefit in this regard.

4 CONCLUSION

In a nutshell our conclusion is that realistic DBMS mechanisms do exist to support the integrity objective of information systems. Some are well established in the literature while others have been proposed more recently and are not so well known. Our principal contribution is to identify these mechanisms and to identify the gaps where none existed or had been fully articulated.

In terms of what DBMS mechanisms can do for us, we can group the nine principles enumerated in this paper as follows.

Group I	Group II	Group III
Well-formed transactions	Least privilege	Authenticated users
Continuity of operation	Separation of duties	Reality checks
	Reconstruction of events	Ease of safe use
	Delegation of authority	

Group I principles are adequately treated by current DBMS mechanisms and have been extensively studied by database researchers. With the single exception of assuring correctness of state transformations these principles can be achieved by DBMS mechanisms. Techniques for implementing well-formed transactions and maintaining continuity of operation across failures have been studied extensively. Their practical feasibility has been amply demonstrated in actual systems. Assuring that well-formed transactions are correct state transformations remains a formidable problem, but there is little that the DBMS can do to alleviate it. As such it is a problem outside the scope of DBMS mechanisms. The DBMS can (i) enforce encapsulation of updates by restricting their occurrence to be within transactions, and (ii) provide controls for installing and maintaining these transactions.

Group II principles need newer mechanisms and conceptual foundations. Several promising approaches have emerged in the literature. Practical demonstration of their feasibility remains to be done, but in concept they do not present prohibitive implementation problems. They do require that current DBMS's be extended in significant ways. Group II principles are the ones where additional DBMS mechanisms hold the promise of greatest benefit.

Group III principles are important but there is little that DBMS mechanism can do to achieve them. Authentication is principally an operating system problem. Reality checks necessarily involve external procedures. Ease of safe use is more an evaluation of the DBMS mechanisms than something to be enforced by the mechanisms themselves. It is facilitated in the DBMS context due to the intrinsic DBMS requirement of user friendly query languages.

In conclusion for group I principles we need little more than has currently been demonstrated in actual products. For group II principles, current systems do something for each one but do not go far enough. There are several promising proposals but no "worked examples." Group III principles are important but are not fully achievable by DBMS mechanisms alone.

References

[1] Burns, R.K. "DBMS Integrity and Secrecy Control." In [18], section A.7, pages 1-4 (1989).

[2] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy*, pages 184-194 (1987).

[3] Clark, D.D. and Wilson, D.R. "Comments on the Integrity Model." In [17], section 9, pages 1-6 (1989).

[4] Clark, D.D. and Wilson, D.R. "Evolution of a Model for Computer Integrity." In [18], section A.2, pages 1-13 (1989).

[5] Codd, E.F. "Extending the Relational Database Model to Capture More Meaning." *ACM Transactions on Database Systems* 4(4): (1979).

[6] Date, C.J. *An Introduction to Database Systems.* Volume I, Addison-Wesley, fourth edition (1986).

[7] Denning, D.E. and Denning, P.J. "Data Security." *ACM Computing Surveys* 11(3):227-249 (1979).

[8] Department of Defense National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria.* DoD 5200.28-STD (1985).

[9] Fernandez, E.B., Summers, R.C. and Wood, C. *Database Security and Integrity.* Addison-Wesley (1981).

[10] Gray, J. "Notes on Data Base Operating Systems." In *Operating Systems—An Advanced Course*, Bayer, R. et al (editors), Springer-Verlag, pages 393-481 (1978).

[11] Gray, J. "Why Do Computers Stop and What Can Be Done About It?" *IEEE Symposium on Reliability in Distributed Software and Database Systems*, pages 3-12 (1986).

[12] Jajodia, S., Gadia, S.K., Bhargava, G. and Sibley, E. "Audit Trail Organization in Relational Databases." In *Database Security III: Status and Prospects*, Spooner, D.L. and Landwehr, C.E. (editors), North-Holland, pages 269-281 (1990).

[13] Linden, T.A. "Operating System Structures to Support Security and Reliable Software." *ACM Computing Surveys* 8(4):409-445 (1976).

[14] Moffett, J.D. and Sloman, M.S. "The Source of Authority for Commercial Access Control." *Computer* 21(2):59-69 (1988).

[15] Murray, W.H. "Data Integrity in a Business Data Processing System." In [17].

[16] Murray, W.H. "On the Use of Mandatory." In [17].

[17] *Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS)*, Katzke, S.W. and Ruthberg, Z.G. (editors), NIST, Special Publication 500-160 (January 1989).

[18] *Report of the Invitational Workshop on Data Integrity*, Ruthberg, Z.G. and Polk, W.T. (editors), NIST, Special Publication 500-168 (September 1989).

[19] Saltzer, J.H. and Schroeder, M.D. "The Protection of Information in Computer Systems." *Proceedings of IEEE* 63(9):1278-1308 (1975).

[20] Sandhu, R.S. "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes." *Journal of ACM* 35(2):404-432 (1988).

[21] Sandhu, R.S. "Transaction Control Expressions for Separation of Duties." *4th Aerospace Computer Security Applications Conference*, pages 282-286 (1988).

[22] Sandhu, R.S. "Transformation of Access Rights." *IEEE Symposium on Security and Privacy*, 259-268 (1989).

[23] Sandhu, R.S. "Mandatory Controls for Database Integrity." In *Database Security III: Status and Prospects*, Spooner, D.L. and Landwehr, C.E. (editors), North-Holland, pages 143-150 (1990).

[24] Selinger, P.G. "Authorization and Views." In *Distributed Data Bases*, Draffan, I.W and Poole, F. (editors), Cambridge University Press, pages 233-246 (1980).

[25] Snyder, L. "Formal Models of Capability-Based Protection Systems." *IEEE Transactions on Computers* C-30(3):172-181 (1981).

[26] Wimbrow, J.H. "A Large-Scale Interactive Administrative System." *IBM Sys. J.* 10(4):260-282 (1971).

A TAXONOMY OF INTEGRITY MODELS, IMPLEMENTATIONS AND MECHANISMS

J. Eric Roskos
Stephen R. Welke
John M. Boone
Terry Mayfield

Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311
(703) 845-3500

1. Introduction

The issue of computer system integrity has become increasingly important as government and industry organizations have grown more dependent on complex, highly interconnected computer systems. A small number of formal models have been established with differing approaches to capturing integrity needs. Based on these models, several model implementations have been developed as possibilities for real systems. In addition, a wide variety of mechanisms exist, independent of these models and model implementations, for addressing particular integrity concerns. These mechanisms implement a set of distinct policies, although the policies are not formally defined as they are in the models. In order to gain a better understanding of what is required to solve integrity problems, this paper reviews a representative set of models, implementations, and mechanisms to identify which specific problems they address. The intent of this paper is not to present recommendations, but rather to identify research topics and existing techniques that apply to computer system integrity.

2. Definition of Integrity

There is currently a lack of consensus in the research community on the definition of integrity. [1] For purposes of this paper, we will adopt two related definitions, as follows.

1. The state that exists when computerized data is the same as that in the source documents and has not been exposed to accidental or malicious alteration or destruction. [2]
2. The state that exists when the quality of stored information is protected from contamination or degradation by information of lower quality. [3]

3. Integrity Models

We describe four integrity models that suggest different approaches to achieving computer integrity. The four models are identified by their author(s): Biba, Goguen and Meseguer, Sutherland, and Clark and Wilson. Not all of these are traditionally viewed as integrity models, yet each on closer examination has specific applicability to integrity. These models establish mathematical definitions that restrict the manipulation of data to achieve the goals of integrity, or formally describe one or more aspects of integrity goals.

3.1 Biba Model

The model defined in [4] was the first of its kind to address the issue of integrity in computer systems. This approach is based on a hierarchical lattice of integrity levels. The goal of this model is to guarantee that every subsystem will perform as it was intended to perform by its creator. A subsystem is some subset of a system's subjects and objects isolated on the basis of function or privilege. The Biba model supports four different integrity policies: the Low-Water Mark Policy, the Low-Water Mark Policy for Objects, the Ring Policy, and the Strict Integrity Policy. Of the four integrity policies, the Strict Integrity Policy is by far the most widely accepted and most familiar, so much so that this policy is often assumed when the Biba model is discussed. Each policy is described below.

The work reported in this paper was conducted as part of Institute for Defense Analyses Project T-Z5-459 under Contract No. MDA903-89-C-0003 for the Department of Defense. It is based on portions of IDA Paper P-2316, *Data Integrity in Department of Defense Computer Systems*, which is in preparation at this time. The publication of this paper does not indicate endorsement by the Department of Defense or the Institute for Defense Analyses, nor should the contents be construed as reflecting the official positions of those organizations.

3.1.1 Low-Water Mark Policy

In this policy, the integrity level of a subject is not static, but is a function of its previous behavior. The policy provides for a dynamic, monotonic and non-increasing value of $il(s)$, the integrity level for subject s. The value of $il(s)$, at any time, reflects the low-water mark of the previous behavior of the subject. The low-water mark is the least integrity level of an object accessed for observation by the subject.

3.1.2 Low-Water Mark Policy for Objects

In addition to changing the integrity level of subjects at each observe access, the Low-Water Mark Policy for Objects also changes the integrity level of objects at each modify access. This alternate policy can be characterized by two rules. First, for each observe access by a subject s to an object o: $il'(s)=\min\{il(s),il(o)\}$ Second, for each modify access by a subject s to an object o: $il'(o)=\min\{il(s),il(o)\}$

3.1.3 Ring Policy

The Ring Policy provides kernel enforcement of a protection policy addressing direct modification. The integrity levels of both subjects and objects are fixed during their lifetimes and only modifications of objects of less than or equal integrity level are allowed. Flexibility of the system is substantially increased by allowing observations of objects at any integrity level.

3.1.4 Strict Integrity Policy

The Strict Integrity Policy is the formal dual of the most common and most thoroughly studied computer security policy model (Bell and La Padula). It consists of three parts: a Simple Integrity Condition, an Integrity *-property, and an Invocation Property. The Simple Integrity Condition states that a subject cannot observe objects of lesser integrity. The Integrity *-property states that a subject cannot modify objects of higher integrity. The Invocation Property states that a subject may not send messages to subjects of higher integrity. Invocation is a logical request for service from one subject to another. Since the control state of the invoked subject is a function of the fact that the subject was invoked, invocation is a special case of modification. Therefore, this rule follows directly from the Integrity *-property.

3.2 Goguen and Meseguer Model

Goguen and Meseguer [5] introduce an approach to secure systems that is based on automata theory. Their approach is divided into four stages: first, determining the security needs of a given community; second, expressing those needs as a formal security policy; third, modeling the system which that community is (or will be) using; and last, verifying that this model satisfies the policy. The authors develop a set theoretic model which is a form of generalized automaton, called a "capability system." It includes an ordinary state machine component, and a capability machine component which keeps track of what actions are permitted to what users.

The Goguen and Meseguer approach is based on the concept of *noninterference*, where "one group of users, using a certain set of commands, is *noninterfering* with another group of users if what the first group does with those commands has no effect on what the second group of users can see." [5] They introduce protection domains to achieve this separation. Noninterference can be extended to address integrity by using the protection domains to ensure that what one group of users does with a certain set of commands has no effect on the data belonging to a second group.

3.3 Sutherland Model

Sutherland [6] presents a model of information that addresses the problem of inference (e.g., covert channels). Sutherland uses a state machine as the basis of his model, but he generalizes the model, apparently to avoid limiting it to the semantic details of one particular type of state machine. Thus, his state machine consists of (1) a set of states; (2) a set of possible initial states; and (3) a *state transformation* function mapping states to states. For each possible initial state, there is an execution sequence defined for each sequence of possible state transformations starting from that initial state. Sutherland generalizes the state machine's execution sequences as a set W of all such execution sequences, which Sutherland terms "possible worlds." A given execution sequence or "possible world" is denoted w, where $w \in W$.[1]

[1] The "possible world" of Sutherland's model is actually even more abstract than we have represented here. As illustrated in a "state machine instantiation" appearing later in [6] under one interpretation of the model, an element w of the set W may consist not only of states, but of "signals" (analogous to the "requests" and "decisions" of the Bell and La Padula model) interspersed between the states.

Sutherland formally represents the information obtainable from a subsequence of w by defining a set of *information functions*, f_i. Each f_i represents the information that can be obtained from one "view" of w. For example, assume a user has full access to the subsequence of w needed to compute $f_1(w)$. If this user knows of an interdependence between f_1 and another information function, f_2, to which the user may have been prohibited access, the user can infer at least some information about $f_2(w)$ from the user's knowledge of $f_1(w)$.

We formalize Sutherland's presentation of this inference as follows: (1) the user knows $f_1(w)=x$; (2) the user deduces $w \in S$ where $S=\{y \mid f_1(y)=x\}$; (3) the user deduces $f_2(w) \in T$ where $T=f_2(S)$; so (4) if there is an interdependence between f_1 and f_2 such that $\exists z \in f_2(W)[z \notin T]$ the user deduces $f_2(w) \neq z$. Steps 1–3 are straightforward generalizations from what the user knows of f_1; the f_2 in steps 1–3 could be an arbitrary function, and calling it f_2 in steps 1–3 only anticipates step 4.

It is in step 4 that the user makes the significant inference. The inference results specifically from the user's knowledge that when the result of a particular subsequence of states is seen, it is impossible for the system to have produced the result z. In such a case, the user is able to conclude that the system has not produced a particular result to which the user may have been denied direct access. The extent to which this inference is useful depends on how much information is represented by knowing that result z was not produced. If the user knows that only two results are possible, knowing "not z" would be of considerable value. Since the Sutherland model is so general, it does include this and similar inferences.

Sutherland goes on to prove a theorem identifying cases in which inference is possible (specifically, cases in which f_1 and f_2 are dependent), and from this theorem derives an important corollary: that information flows are always symmetric. This corollary has importance to integrity since it shows that the user who can control the computation of f_1 can influence the result of f_2. This situation can be thought of as a reverse covert channel.

3.4 Clark and Wilson Model

Clark and Wilson [7,8] make a distinction between what they term "military security" and "commercial integrity," and present a model for achieving data integrity. They emphasize three assertions. First, they claim that there is a distinct set of security policies, related to integrity rather than disclosure, applicable to computer systems. Second, they maintain that integrity policies are often of higher priority than nondisclosure policies in the commercial data processing environment. Finally, they maintain that separate mechanisms are required for enforcement of these policies, disjoint from those required by the Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) [2].

There are two keys to the Clark and Wilson integrity model: the well-formed transaction, and separation of duty. A *well-formed transaction* is structured so that a user can not manipulate data arbitrarily, but only in constrained ways that preserve or ensure the *internal consistency* of the data. *Separation of duty* attempts to ensure the *external consistency* of data objects: the correspondence between a data object and the real world object it represents. This correspondence is ensured indirectly by separating all operations into several subparts and requiring that each subpart be executed by a different person.

The Clark and Wilson model is defined in terms of four elements: constrained data items (CDIs), unconstrained data items (UDIs), integrity verification procedures (IVPs), and transformation procedures (TPs). CDIs are data items within the system to which the integrity model must be applied. UDIs are data items not covered by the integrity policy that may be manipulated arbitrarily, subject only to discretionary controls. New information is fed into the system as a UDI, and may subsequently be transformed into a CDI.

The TPs and IVPs are two classes of procedures that implement the concept of a well-formed transaction. The purpose of TPs is to change the set of CDIs from one valid state to another. The purpose of an IVP is to confirm that all of the CDIs in the system conform to the integrity specification at the time the IVP is executed. An IVP checks internal data consistency, and also may verify the consistency between CDIs and external reality.

Clark and Wilson achieve separation of duty through what they term "access triples." Whereas the lattice model of Bell and La Padula defines access restrictions for subjects to objects, the Clark and Wilson model partitions objects into programs and data and requires subject/program/data triples. Separation of duty is discussed further in a later section of this paper.

4. Integrity Model Implementations

Implementations suggest realistic approaches to the theoretical basics identified by models. Seven methods have been identified as implementations of one or more of the fundamental models described in the previous section. The seven implementations are: Lipner, Boebert and Kain, Lee, Shockley, Karger, Jueneman, and Gong.

4.1 Lipner Implementation

Lipner [9] examines two ways of implementing integrity in commercial data processing systems. The first method uses the Bell and La Padula security lattice model by itself. The second method combines the Bell and La Padula model with Biba's integrity lattice model. Lipner's approach requires looking at security requirements in a different way from the prevalent view taken in the national security community. In particular, non-hierarchical categories are considered more useful than hierarchical levels.

Lipner makes the important early distinction of separating objects into data and programs. Programs can be either passive (when they are being developed or edited) or active (when they are being invoked on behalf of a particular user), while data objects are always passive. Programs are the means by which a user can manipulate data; thus, it is necessary to control which programs a user can execute and which data objects a program can manipulate. Also, Lipner emphasizes that special attention must be given to programs to ensure that they perform as expected and do nothing more.

4.2 Boebert and Kain Implementation

Boebert and Kain [10] introduce an implementation of the Goguen and Meseguer model based on the Honeywell LOCK (formerly SAT) machine. Their implementation provides a mechanism for ensuring that data of particular types can only be handled by specific trusted software. Boebert and Kain's implementation is an object-oriented, capability-based approach that focuses more on isolating the action than isolating the user. Domains restrict actions to being performed in only one place in only one way; if you don't have access to that domain, you can't perform that action.

4.3 Lee and Shockley Implementations

Independently, Lee [11] and Shockley [12] developed implementations of the Clark and Wilson model using Biba integrity categories and partially trusted subjects. Both implementations are based on a set of sensitivity labels. This set is built from two essentially independent components: every label contains a pair of sensitivity markings: one element of the pair represents a sensitivity with respect to disclosure, and the other represents a sensitivity with respect to modification. Mathematically, the labels have a dominance relation that partially orders them, such that least upper and greatest lower bounds are uniquely defined. These bounds are used as a means of restricting access to programs and data.

The Lee and Shockley implementations make two important suggestions. First, categories can be interpreted as data types; thus, strong typing is important for implementing integrity. Second, a transaction-oriented approach works well; thus, database technology should be very relevant to integrity.

4.4 Karger Implementation

Karger [13] proposes an implementation of the Clark and Wilson model which combines SCAP, the author's secure capability architecture [14] with the lattice security model. In this scheme, a capability-based protection kernel supports access control lists (representing the security lattice). In a typical capability system, the processor automatically loads capabilities into a cache for quick reference. Karger's proposal is to cause a fault or trap to the most privileged software domain (i.e., the security kernel) on the first reference to a capability, when it would normally be moved into the cache, so that this privileged domain can evaluate whether the lattice permits the capability to be used. Once a capability has been evaluated, it is placed in the cache so that it does not have to be reevaluated. If a lattice entry is changed, revocation can be achieved by flushing all of the capabilities for that object from the cache (causing new requests for the object to be freshly evaluated).

As part of this implementation, audit trails form a much more active part of security enforcement than in earlier systems. Karger introduces *token capabilities* to make the use of audit information easier. While taking the form of capabilities to prevent unauthorized tampering, token capabilities are in fact separate copies of individual audit records. Token capabilities are used in conjunction with access control lists to ensure that permission to execute certain transactions can only be granted if certain previous transactions have been executed by specific individuals.

4.5 Jueneman Implementation

Jueneman [15] proposes a defensive detection scheme that is based on mandatory and discretionary integrity controls, encryption, checksums, and digital signatures. This approach is intended for use on dynamic networks of interconnected trusted computing bases (TCBs) that communicate over arbitrary non-secure media. Mandatory Integrity Controls *prevent* illegal modifications within the TCB, and *detect* modifications outside the TCB. Discretionary Integrity Controls are used in supplementing the mandatory controls to *prevent* the modification, destruction, or renaming of an object by a user who has the necessary mandatory permissions, but is not authorized by the owner of the object. Encryption is used by the originator of an object to protect the secrecy or privacy of information. Checksums provide immutability and signatures provide attribution to allow the recipient of an object to determine its believability. The *originator* of an object should be responsible for assuring its confidentiality. The *recipient* should be responsible for determining its integrity.

4.6 Gong Implementation

Gong [16] presents the design of an identity-based capability protection system (ICAP), which is intended to be used on a distributed system in a network environment. ICAP merges the access control list approach and the capability approach; access control lists support a capability protection mechanism (the opposite of Karger's SCAP), and thereby solve the problem of revocation. Gong's design provides support for administrative activities such as traceability. This approach also works for a centralized system. Compared with existing capability system designs, ICAP requires much less storage and has the potential of lower cost and better real-time performance.

5. Integrity Policies, Principles, and Mechanisms

Although a wide variety of integrity mechanisms exist, on examination they are found to serve a relatively small set of distinct purposes. In this paper, we use the term *policies* to describe the higher-level purposes of mechanisms, since these purposes generally reflect administrative courses of action devised to promote integrity. Some of the policies intended to promote integrity are familiar to individuals acquainted with the traditional computer security literature, while others are less so. Some of the mechanisms we will examine appear primarily in embedded systems, and some are not found in familiar software systems, but are used in specialized application environments such as accounting. We will identify these policies, and the mechanisms that enforce them, in the following discussion. We begin with some of the simplest policies and mechanisms, and proceed to more complex ones. In several of these policies, there are underlying *principles* which some or all of the mechanisms employ in implementing the policy. Where such principles are evident, we will present the principles before discussing the mechanisms.

5.1 Policy of Authorized Actions

Many integrity mechanisms are based on a policy of *authorized actions*, which specifies that users may only perform those actions for which the users are authorized. This policy is so central to computer security that at first it might seem that all integrity mechanisms would implement this policy, and indeed it is the case that many mechanisms at least require a policy of authorized actions to be effective. However, we characterize mechanisms as primarily implementing this policy if the mechanism is directly concerned with the goal of arbitrarily-specified access control, rather than with some other purpose which requires a policy of authorized actions for its interpretation. There are two underlying principles employed in implementing this policy, deterrence and prevention of prolonged access. Among the mechanisms that implement this policy are those discussed in sections 5.1.3 through 5.1.7.

5.1.1 Principle of Deterrence

An ideal goal of computer security mechanisms is to provide absolute controls; in the implementing of a policy of authorized actions, such a goal would require that unauthorized actions not be possible to perform. Where integrity is concerned, this is not always possible. If a policy of authorized actions states that "a user who is otherwise authorized for specific actions shall not be authorized to employ those authorized actions in unethical ways," absolute controls may not be possible, since what is ethical vs. unethical may be a matter of human judgement. In such a case, it may be necessary that the principle of *deterrence* be employed. This principle is based on the idea that many people may be deterred from performing certain actions by making such actions sufficiently difficult or unpleasant. Deterrence may involve the risk of being caught performing an unauthorized action, or may simply involve making the unauthorized action sufficiently difficult to perform that the desired result is not worth the effort. Clearly, more absolute controls are preferable in most cases, but it must be emphasized that absolute controls are not always possible, particularly in integrity.

5.1.2 Principle of Prevention of Prolonged Access

In cases in which absolute controls are not possible, it is sometimes useful to employ the principle of *prevention of prolonged access*. According to this principle, the likelihood of integrity violations taking place may be reduced by reducing the amount of time an individual is given access to data or resources. This principle is weaker than deterrence, in that it assumes the subject will not have sufficient time to perform improper modifications of data, or will not be able to discover exploitable weaknesses in the available time.

5.1.3 Unconditional Authorization

The simplest class of mechanisms implementing this policy is one which directly grants or denies authorization for a particular action. *Access control lists* (ACLs) are an example of this class.

5.1.4 Conditional Authorization

Conditional authorization mechanisms grant authorization only when specified conditions are satisfied. One example is an "emergency override" mechanism, which allows prohibited actions when an emergency condition occurs. These mechanisms are also present in military systems in which authorization to perform certain actions is allowed only when a combat action is in progress.

5.1.5 Access Keys

The class of mechanisms involving *access keys* is based on the subject's possessing a data object or being assigned a data value which is tested to determine the subject's authorization to modify data. One of the simplest forms of this mechanism appeared in the memory management units of early central processing units (CPUs), where each memory segment or page was assigned a "key" which would be matched against a key stored into a protected CPU register during a context switch. A memory reference would succeed only if the key on the memory segment matched the key in the protected register. This same mechanism is found in present-day file systems which grant permission to perform certain operations only to the "owner" of a file, since the user-ID of the process attempting to perform the operation on the file can be considered to be a key which must be matched against the owner-ID for the file.

The access key mechanisms discussed above are based on a test of equality. Other tests are possible, such as a test under a partial ordering relation. An example is the mandatory access control labelling prescribed by the *Trusted Computer Systems Evaluation Criteria* (TCSEC) [2], or the integrity labels required by the Strict Integrity Policy of the Biba model [4]. These access control mechanisms also fall under the class of access keys.

5.1.6 Value Checks and Range Checks

Value checks and *range checks* serve as a weak mechanism for detecting improper data at the time it is entered or processed; the goal of these checks is primarily to detect errors that would result in accidental entry of incorrect data. The data are checked to be sure they have one of a specific set of values, or that the values fall into a specific range or set of ranges. Although simple, these checks are noteworthy in that they are one of the types of checks that are most often moved by *vertical migration* from the software level to the level of microcode or firmware. They include checks that data have values appropriate to specific primitive data types, such as the check that a packed decimal number does not contain the hex digits A-F and that its sign bits have one of the appropriate values. We categorize the mechanisms as implementing a policy of authorized actions since the purpose of the checks is often to ensure that input data is in an authorized format or has authorized values, although it may be correctly argued that other policies are also supported by these mechanisms.

5.1.7 Access Control Tuples

Access control tuples provide one of the most direct ways of implementing a policy of authorized actions. They consist simply of tuples that indicate subjects and the actions they are authorized to perform on specified objects. In implementations of systems satisfying the TCSEC, these usually take the form of ACLs. In the Clark and Wilson model [7], they take the form of a triple specifying a user, transformation procedure, and object. When implemented, one element of the tuple may be omitted, being implied by the entity to which it is attached or which possesses it (subjects, in the case of capabilities; objects, in the case of of ACLs).

5.2 Policy of Supervisory Control

A *policy of supervisory control* requires that a subject in a specific *supervisor* role be required to authorize specific actions as they arise. This policy may involve either requiring the supervisor to give final approval to an action that was decided upon by another individual, or requiring the supervisor to give approval only to actions that meet certain constraints, such as actions having a particularly large scope or effect.

5.2.1 Principle of Alarm Sufficiency

In interpretations of a policy of supervisory control, it is critical that communications channels that indicate requests for action to the supervisor not be blocked by excessive message traffic. The *principle of alarm sufficiency* requires that, to ensure integrity of data, alarms (the channels by which requests are transmitted to the supervisor) must be sufficient to support all possible combinations of alarms that can be active at once. This principle is an integrity concern since, if alarm sufficiency is not enforced, it would be possible to mask a given item of data requiring a supervisory decision by simultaneously causing other data to be transmitted to the supervisor.

5.2.2 Mechanism of Supervisory Authorization

Supervisory control is a straightforward policy to implement, and mechanisms that implement it can be placed in the category of *supervisory authorization* mechanisms. Examples of such mechanisms are the requirement that a bank officer approve withdrawals that exceed a certain amount, or that a fire control officer approve firing a missile upon an aircraft believed to be hostile during a military engagement.

5.3 Policy of Separation of Duty

In some environments, it is more likely for an integrity violation to occur when operations are carried out by individuals working alone. This can be because there is less likelihood of detection in such a case, or because there is not a second person to notice accidental errors. In well-controlled environments, it may be possible to partition a task among several individuals such that no one individual has enough information or control available to successfully carry out an integrity violation without allowing detection: the person will not be able to "cover his tracks." In such an environment, a policy of *separation of duty* may be employed to prevent an individual from acting alone to carry out an integrity violation, by making it essential that two or more people be involved in an operation. With separation of duty in effect, it is necessary for two or more people to conspire together to carry out an integrity violation, something which usually is less likely, and may be more easily detectable in subsequent investigation, than an integrity violation carried out by one person acting alone.

5.3.1 Conventional Access Control

The conventional access control tuples discussed in the previous section may be used to implement separation of duties, by appropriate assignment of access permissions. Appropriate assignment of access via the access control tuples must be made to ensure that access permissions for two separated duties are not assigned to the same person.

5.3.2 Compartmentation

Traditional compartmentation of information, similar to that used in implementing confidentiality policies, can be used to effect separation of duties, via mechanisms such as those in [4]. Compartmentation generally requires fine-grained definitions of categories, with separate categories for duties intended to be separated, and non-overlapping assignments of these categories to individuals.

5.3.3 Chinese Wall

A mechanism used in some financial institutions is the *Chinese wall*, also called the Brewer-Nash policy [17]. In this mechanism, data or operations are grouped into equivalence classes. Each subject is originally given access to all data or operations, but access to one element of an equivalence class by a subject causes access permissions to change such that the subject cannot access other data or operations in the same equivalence class. Thus, access permissions are dynamically reconfigured to ensure separation. Dynamic reconfiguration automates the assignment of access permissions to ensure separation of duty. It should be noted that this mechanism was termed a "policy" by Brewer and Nash, although we here categorize it as a mechanism implementing a policy of separation of duty.

5.3.4 *N*-Person Control

A different form of separation of duty is provided by mechanisms involving *n*-person control. In such mechanisms, *n* people must simultaneously request or authorize an operation in order for it to take place. The principal purpose of such a mechanism is to ensure that a single errant individual or system component cannot cause a critical process to be authorized or initiated.

5.3.5 Process Sequencing

To further organize separation of duty, *process sequencing* is sometimes used. This approach requires that duties not only be separated, but also be performed in a specified order. Doing so can ensure that individuals assigned duties later in the sequence will always check the work of their predecessors, or that sufficient information is always available for each individual to perform their assigned duty correctly. An example of the absence of process sequencing would be a situation in which a person signed a check before the amount to be written on the check was computed.

5.4 Policy of Rotation of Duty

Where separation of duty is employed, increased integrity may sometimes result from an additional policy of *rotation of duty*. The purpose of such a policy can be to reduce errors that result from repetition of a monotonous task, or to reduce the likelihood that an individual will discover or exploit ways of carrying out an integrity violation. Again, this policy is useful primarily in cases where absolute integrity controls are not possible. The mechanisms are the same as those for separation of duty, except that a way of atomically rotating duties must be provided so that exploitable intermediate states do not exist while the rotation is taking place.

5.5 Policy of Separation of Resources

In addition to separating duties among different persons or processes, access to resources can also be separated. In *separation of resources*, the resources to which subjects have access are partitioned such that a given subject has access only to a subset of the resources available. More precisely, a given subject *and the tools available to that subject* are allowed access only to specific resources. This policy can also overlap the policy of separation of duties if controlling access to resources also limits the duties which each subject can perform.

5.5.1 Capabilities

Capabilities are specially-protected objects, or values representing such objects, the possession of which gives access to other objects to which access is to be controlled. The capability is used to name the object to which it refers, and usually has specific attributes or permissions associated with it which specify what types of access the possessor of the capability is granted. It is possible for a system to invalidate a capability, so that possession of it no longer grants access; to control whether or not it is possible to give away the capability to another process; or to limit the number of times a capability may be used to access an object [18].

5.5.2 Descriptors

Whereas capabilities are associated (by possession) with a subject, *descriptors* are associated with an object and indicate which subjects are granted access to the object. A variety of memory protection schemes based on descriptors have been developed over the history of computer architecture, since descriptors are a very old and widely-used mechanism. Most of these mechanisms work by comparing addresses generated by a user program with a list of valid address ranges which the program is allowed to access. Before an attempted access is permitted to occur, the comparison is performed, and if it is found that the address is illegal, an exception is generated which aborts the access.

5.5.3 Separation of Name Spaces

A more primitive mechanism for separation of resources is *separation of name spaces*. In this mechanism, two distinct user processes are given separate address spaces, such that the same address refers to distinct, non-overlapping locations when used by each process. The result is that each process is unable to modify the other's data not because a fault is generated which prevents such an access, but because the address has a different meaning to the two processes. The mechanism also appears in simple "virtual machine" interpretations in which each user appears to have access to a private machine which is not shared with other users.

5.6 Policy of Encapsulation

Protection against modification of data can be increased by encapsulating distinct parts of a system into *objects*[2] and controlling the ways in which individual objects in a system may be accessed. There two underlying principles employed in implementing encapsulation, abstract data types and strong typing.

5.6.1 Principle of Abstract Data Types

To precisely define the semantics of data and to control the operations that may be performed on them, the principle of *abstract data types* is employed. This principle defines a *type* to be a particular set of data values and a particular set of operations that may be performed on those values.

5.6.2 Principle of Strong Typing

The principle of *strong typing* is a major integrity mechanism, and is one of the bases of the Clark and Wilson model of data integrity. Strong typing is simply the strong, uncircumventable enforcement of abstract data types. It restricts what operations can be performed on what data (or on what objects). When strong typing is enforced, even though an object's representation might be compatible with operations not associated with that data type, such operations will not be permitted. This principle is distinct from simple abstract data typing, in which data types are defined, but may be circumvented when convenient for the programmer to do so.

[2] The term *object* as used in this discussion is distinct from the term *object* as generally used in discussions of computer security.

5.6.3 Message Passing and Actors

If the only way to access objects is by sending *messages* to them, the types of actions that can be performed on the objects are restricted to those which the object directly permits, since the only way to perform such actions is to request the object to perform an operation on itself. This approach is the basis of *message passing* encapsulation mechanisms. A well-developed form of this mechanism is the *actor* model [19]. It is also found in the Smalltalk programming language [20].

5.6.4 Data Movement Primitives

Most computer architectures equate a data object with the place in which it is stored. A computer architecture which uses the *data movement primitives* [21] **get** and **put** distinguishes these two; an object has a distinct identity separate from its address, and it is possible to move this object about, removing it from one address and placing it in another. This mechanism addresses integrity problems related to *multiple updaters*.

5.7 Policy of Fault Tolerance

A policy of *fault tolerance* specifies that a system should attempt to survive isolated failures of components. Policies of fault tolerance potentially involve two parts. The first is the detection of errors, and is commonly accomplished via *summary integrity checks*. This part is necessary for a system to determine when a failure has occurred, since a system that simply corrects errors without detecting them can be considered not to have failed at all. The second is the attempted correction of errors. This part is accomplished via the *principle of redundancy*.

5.7.1 Principle of Redundancy

Redundancy is commonly used in fault-tolerant applications. The same data or processes are duplicated several times, possibly separated in time or location, in the expectation that not all of the redundant copies will fail in the same way or at the same time. *Hardware redundancy* is the most familiar type of redundancy, and involves duplicated hardware components. *Software redundancy* involves additional software beyond what is necessary for basic operation, in order to check that the basic operations being performed are correct. *Information redundancy* involves duplication of information, possibly using different encodings or representations of the information to reduce the likelihood that the same failure will modify all copies in the same way. *Time redundancy* involves repeating an operation at several separated points in time, to detect intermittent or transient failures [22].

5.7.2 Summary Integrity Checks

Summary integrity checks are mechanisms which "summarize" the content or state of data, such that this summary can be checked against the data itself to ensure that the data has not been modified. To be effective, the summary information must be separated from the data summarized, and independently protected, or must be computed in a way that cannot easily be duplicated. Summary integrity check mechanisms include *transmittal lists*, which simply list the contents of an aggregate data object; *data counts*, which count the size or number of data objects; *checksums* and *hash totals*, which are abbreviated functions of a larger block of data which they represent; and *check digits*, which are simply a short, single-digit checksum.

6. Conclusion

Four models have been developed which suggest fundamentally different ways of achieving computer integrity. Biba's [4] was the first model to address integrity in a computer system, and it has the most established background because it is based on the Bell and La Padula model for confidentiality. This model emphasizes the use of hierarchical levels. Goguen and Meseguer [5] uses domain separation and automaton-based state transformations. Sutherland [6] focuses on ensuring that conceptually independent computations do not have unintended interdependencies. Clark and Wilson [7,8] introduce the concepts of well-formed transaction and separation of duty; separation of duty is implemented by access triples.

Based on these four models, seven model implementations have been identified and described. Lipner's implementation [9] is the first work to emphasize the importance of non-hierarchical categories for achieving integrity. This implementation also introduces the important distinction between program objects and data objects. Boebert and Kain [10] describe a flexible, object-oriented, capability-based approach that focuses more on isolating the action than isolating the user. Lee [11] and Shockley [12] introduce minimum and maximum views for controlling access to objects. This approach points out the usefulness of strong typing and transaction-based operations. Karger [13] and Gong [16] both combine the advantages of capabilities (speed and domain separation) and access control lists (review and revocation). Karger's approach uses capabilities in support of ACLs; Gong uses ACLs in support of capabilities. Jueneman [15] uses encryption, checksums, and digital signatures to achieve integrity in a distributed system.

In examining integrity mechanisms, we find that many mechanisms exist, but that they ultimately implement a relatively small set of distinct policies. The general *policy of authorized actions* specifies that users may perform only those actions for which they are authorized; this is a concept that is as fundamental to confidentiality as it is to integrity. The *policy of supervisory control* classifies actions according to their severity of impact, reserving the actions with the greatest impact to be performed or confirmed by more-trusted individuals serving in a *supervisor* role. The *policy of separation of duty* requires that duties be divided across two or more individuals, such that no one person can complete a controlled action while acting alone. The similar *policy of rotation of duty* requires that individuals be rotated among the set of duties, reducing the duration of exposure of a given individual to a particular duty. The *policy of separation of resources* controls access to resources needed to perform an action, rather than directly controlling which actions may be performed. The *policy of encapsulation* encapsulates a system into objects which may be accessed in controlled ways, thus restricting the ability for "back door" accesses to system components, and increasing the amount of control the component can exercise over what actions are performed. A *policy of fault tolerance* requires that isolated faults and errors, when they occur, be detected and corrected.

No single model, taken on its own, supports all of the policies. Yet, the models as a whole can be argued to provide at least a moderate level of support for at least some interpretation of these policies. Clearly, addressing all the aspects of integrity requires an integration of model concepts to ensure complete coverage, or selection of models appropriate to the specific policies to be enforced.

References

[1] Ruthberg, Z.G. and W.T. Polk, eds., *Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS)*, NIST Special Publication 500-160, October 1987.

[2] Department of Defense, *DoD Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, Washington, DC, December 1985.

[3] Courtney, R.H., Jr., "Some Informal Comments About Integrity and the Integrity Workshop," in *Report of the Invitational Workshop on Data Integrity*, NIST Special Publication 500-168, pp. A.1.1-A.1.18, January 1989.

[4] Biba, K.J., "Integrity Considerations for Secure Computer Systems," ESD-TR-76-372, Bedford, MA: MITRE Corporation, April 1977.

[5] Goguen, J.A. and J. Meseguer, "Security Policies and Security Models," in *Proceedings of the 1982 Berkeley Conference on Computer Security*, pp. 11-20, 1982.

[6] Sutherland, D.I., "A Model of Information," in *Proceedings of the 9th National Computer Security Conference*, pp. 175-183, September 1986.

[7] Clark, D.D. and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pp. 184-194, April 1987.

[8] Clark, D.D. and D.R. Wilson, "Evolution of a Model for Computer Integrity," in *Report of the Invitational Workshop on Data Integrity*, NIST Special Publication 500-168, pp. A.2.1-A.2.13, January 1989.

[9] Lipner, S.B., "Non-Discretionary Controls for Commercial Applications," in *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pp. 2-10, April 1982.

[10] Boebert, W.E. and R.Y. Kain, "A Practical Alternative to Hierarchical Integrity Policies," in *Proceedings of the Eighth National Computer Security Conference*, pp. 18-27, September 1985.

[11] Lee, T.M.P., "Using Mandatory Integrity to Enforce 'Commercial' Security," in *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pp. 140-146, April 1988.

[12] Shockley, W.R., "Implementing the Clark/Wilson Integrity Policy Using Current Technology," in *Proceedings of the 11th National Computer Security Conference*, pp. 29-37, October 1988.

[13] Karger, P.A., "Implementing Commercial Data Integrity with Secure Capabilities," in *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pp. 130-139, April 1988.

[14] Karger, P.A. and A.J. Herbert, "An Augmented Capability Architecture to Support Lattice Security and Traceability of Access," in *Proceedings of the 1984 IEEE Symposium on Security and Privacy*, pp. 2-12, April 1984.

[15] Jueneman, R.R., "Integrity Controls for Military and Commercial Applications, II," in *Report of the Invitational Workshop on Data Integrity*, NIST Special Publication 500-168, pp. A.5.1-A.5.61, January 1989.

[16] Gong, L., "A Secure Identity-Based Capability System," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 56-63, May 1989.

[17] Brewer, D.F.C. and M.J. Nash, "The Chinese Wall Security Policy," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 206-214, May 1989.

[18] Gligor, V.D., J.C. Huskamp, S.R. Welke, C.J. Linn, and W.T. Mayfield, "Traditional Capability-Based Systems: An Analysis of Their Ability to Meet the Trusted Computer Security Evaluation Criteria," IDA Paper P-1935, Alexandria, VA: Institute for Defense Analyses, February 1987. Available as NTIS AD-B119 332.

[19] Agha, G.A., *Actors: A Model of Concurrent Computation in Distributed Systems*. Cambridge, MA: The MIT Press, 1986.

[20] Goldberg, A. *Smalltalk-80: The Language and its Implementation*. Reading, MA: Addison-Wesley, 1983.

[21] Roskos, J.E., "Data Movement, Naming, and Ambiguity," Vanderbilt University Department of Computer Science Technical Report CS-84-05, March 1984.

[22] Johnson, B.W., *Design and Analysis of Fault Tolerant Digital Systems*. New York, NY: Addison-Wesley Publishing Co., 1989.

Executive Summary

National Computer Security Policy

Lynn McNulty

National Institute of Standards and Technology

This panel will discuss the changing national authorities and policies with regard to computer security and the role that agency computer security policy plays in a successful computer security program. The passage of the Computer Security Act of 1987 and the rescinding of NSDD-145 have resulted in a changed national computer security policy structure. OMB, NIST, NSA, GSA and other agencies are involved in developing governmentwide policies and standards to varying degrees. Policies developed by individual agencies, which build upon policies, standards and guidelines issued by national authorities, plays a critical role in ensuring clear direction to agency executives, managers and system users. The panel will also discuss areas in which national policy and/or standards guidance have been successful and those areas where further attention may be necessary.

A BRIEF TUTORIAL ON TRUSTED DATABASE MANAGEMENT SYSTEMS

John R. Campbell
National Computer Security Center
Office of Research and Development
9800 Savage Road
Fort George G. Meade, Maryland 20755-6000
(301)859-4488

INTRODUCTION

Over ninety percent of the nation's mainframes and most minicomputers and microcomputers contain database management systems (DBMS). Our most critical data, including defense, intelligence, law enforcement, social welfare, and financial data, are stored on such systems. Applications ranging from financial systems to national defense mechanisms depend on the security of these systems.

The building of these systems and the construction of applications for these systems is a multi-billion dollar industry. Yet, to date, little has been done to secure database management systems. Vendors have emphasized performance and ease of use, with security being an afterthought. Often any security included in the database system is done without regard to consistency with the existing operating system security mechanisms.

This lack of interest in DBMS security, however, is starting to change. The threat to data, due to nondisclosure, lack of integrity and unavailability, is beginning to be addressed. Trusted products are being introduced. The National Computer Security Center (NCSC) is leading the research and development, in part, through contracts with Oracle Corporation and Teradata Corporation. These help the government to examine the fundamental issues regarding the Orange Book interpretation of database management systems. Teradata is using a database machine architecture while Oracle is developing host based systems.

This tutorial gives the background, describes the issues and offers some proposed solutions for database security.

DATABASES AND DATABASE SECURITY

In the August 1989 issue of Computer [JACO89], the reviewer of a book on computer security makes two comments, both I especially agree with for database security. First, he states that the entire field of computer security has substantial weaknesses. This is especially true for database security. For example, trusted distributed database management systems present many unanswered questions. There is no general theory of control for inference and aggregation, although there are some application specific controls. Verification tools are weak. There are many other unanswered issues.

Second, the reviewer states that the field of computer security is quickly evolving. Again, this is especially true for database security. It is junior to operating system security because it often has to depend on a trusted operating system. But, until now, there were few trusted operating system products. Several years ago, we talked about the possibility of trusted database systems. Today there are at least eight prototypes, half of which are commercial quality. Truly the field is rapidly evolving.

What is a database? Date [DATE86] defined them as collections "of stored operational data used by the application systems of some particular enterprise." The operational data could include product, account, patient, student or planning data. It does not include input or output data, work queues, temporary results or any purely transient information. Databases are increasing in complexity. The data can now be pictures, rules, or derived information.

What is a database management system? Date [DATE86] defines these as systems that provide users with a view of the database that is elevated somewhat above the hardware level, and support user operations such as SQL operations that are expressed in terms of that higher level view. "SQL", or Structured Query Language, is a high level query language that contains both data manipulation and data definition features. It also contains data control features, "grant" and "revoke", for example. Database management systems are also increasing in complexity. Some database systems have natural language, rule manipulation and other artificial intelligence components. Some are distributed. Database security must meet these challenges.

WHY DATABASE SECURITY IS IMPORTANT

Database security is important because databases are so very important. The DoD, the intelligence, financial, law and social services communities depend on them to be safe and correct. Two billion dollars was spent in 1987 on database systems. It is estimated that six billion will be spent in 1992. Applications for these systems cost many times more. Ninety percent of mainframes use database systems.

Database security is important because even with a trusted operating system underneath, data is at risk if you are not using a trusted database system. One problem is granularity. Operating systems usually protect at the file level. Databases need finer granularity such as table or relation, row or tuple, or even element. Database systems can provide protection at these levels of granularity. In addition, different discretionary security policies are often desired for database systems that restrict access to specific data through specific database operations, such as insert, update, retrieve and delete. Such controls are not available in operating systems.

Database security is important because database systems are the most widely used class of application on computer systems. As such, much learned about database systems, such as trusted operating system interface, can be transferred to our knowledge of securing other applications.

Database security includes data integrity. Data integrity is important because a database is useless if the information you get out of it is wrong. The importance of integrity has long been realized by database system vendors and they have provided some capabilities to preserve integrity. However, the active data dictionary, where data constraints are recorded and enforced, is a relatively new concept.

Concentrated work done now on both database security and integrity is important because the list of problems is constantly growing. In addition to the vanilla stand-alone commercial database systems, which by themselves are quite complex, we now have commercial expert, multimedia and/or distributed database systems. These, plus intelligent, temporal, historical and object-oriented databases add to the complexity of the problem.

SOME ARCHITECTURES AND MODELS

Database systems employ different architectures and these present differing problems. Database machines are computers dedicated to database activities. All data is stored on these machines. Host computers issue queries to the database machine. This machine processes the query, finds and manipulates the data and returns the answer. Under this configuration, the machine's operating system (OS) and database system are usually one; therefore the OS/DBMS interface does not exist.

In host-based DBMSS, the OS/DBMS interface is a serious problem. Here the DBMS runs on a general purpose computer that, in addition to the DBMS, usually has other applications running on it. Some vendors want to port their database systems to as many computers as possible. How is this accomplished in an efficient yet secure manner? There are no standard security interfaces. Therefore, in order to be truly portable, DBMS vendors may choose to duplicate the security functionality of the operating system and not use the security functionality of the operating system. This avoids having to make several custom interfaces, but it increases the complexity and size of DBMS security components. Also, if the DBMS is trusted, its interactions with the operating system trusted computing base must be controlled.

Finally, distributed database systems have added additional complexities to the security problem. The data in these systems may have different physical locations, may be on heterogeneous nodes and may be redundant. How do you audit? How do you identify and authorize? How do you assure the integrity of redundant information? We are beginning to address these issues.

The DBMS model used may also affect security. Is the model relational, network, hierarchical, object-oriented or other. A secure entity relationship study reported that it was easier to secure a system based on an entity relationship model than a relational model. One reason he gave was that he had the freedom to choose the entity-relation model that could best contain security. There is no standard model. The relational model, however, has solidified into almost a standard, a standard where initially security was not considered, and therefore retrofitting security, especially multilevel security, is difficult.

WHAT IS SECURITY?

Security, in some areas, has been equated only with nondisclosure. A system is secure if you can prevent unauthorized users from reading sensitive information. However, we also include integrity and availability or denial of service components in this definition. Consequently, our definition agrees with what the Strategic Defense Initiative calls "security *".

WHAT IS INTEGRITY?

We've seen a list of 150 definitions of integrity. One we like is "sound, unimpaired or perfect condition" [NCSC88a]. Is what you get out of the database what you put in it?

Three integrity components have been noted. The Department of Defense Trusted Computer Evaluation Criteria (TCSEC or "Orange Book") [DOD85] recognizes two types, label integrity and system integrity. Label integrity assures that the security labels accurately represent the classifications of subjects or objects with which they are associated. System integrity is the correct operation of the on-site hardware and firmware elements of the TCB. This "TCB" is the totality of protection mechanisms within a computer which is responsible for enforcing a security policy.

What the TCSEC doesn't explicitly mention, the third integrity component, data integrity, is something very important to DBMS users. We define it as the "property that data has not been exposed to accidental or malicious alteration or destruction [NCSC88b].

DATA INTEGRITY IMPLEMENTATION

Data integrity may be implemented as part of the overall security policy. For example, the Biba integrity model [BIBA77] may be implemented with Bell-LaPadula nondisclosure model [BELL73] to produce a model that enforces both integrity and security. SeaView did this using a modified Biba model and Bell-LaPadula. The model can then be translated into an operational system.

Even though a security policy may not be explicitly stated, integrity components may exist. Entity integrity, for example, does not permit null primary keys. In general, under referential integrity, foreign keys must reference existing primary keys. Also, integrity constraints and typing may be used. For example, one field or attribute may allow only months of the year, with the first letter capitalized. The system will check that each item entered into this field satisfies these constraints. Both secure recovery and the concept of serializability are also important for data integrity.

Finally, it is important to note that nondisclosure and data integrity may conflict. Referential integrity may enable someone at a lower classification level to know whether something at a higher level exists. Hiding the existence of high data from low users may also require that polyinstantiation be used. Under this concept, multiple data objects with the same name, differentiated by their access class, may exist simultaneously [DENN88]. Is this an integrity violation? And couldn't it cause data integrity problems?

BREAKDOWN OF THE PROBLEMS

It is useful to break down the database security problem into historical components. Research that has been done in each of these components may be useful in building a secure database system.

The first component is operating system security. Many of the concepts that originated in operating system security are also used in DBMS security. In addition, in the computer system, the DBMS may be layered on top of the OS, may depend on the OS for services and may share the responsibility for security policy enforcement with the OS.

The second component is network security. Network security concepts will be useful in distributed database work.

Some issues, such as granularity, are unique to database security. The problems of inference and aggregation, while not unique to database systems, are exacerbated by database management systems, because these systems are designed to easily manipulate large quantities of data. Finally, there are issues unique to the distributed DBMS.

STANDARDS/INTERPRETATIONS

Several useful standards and interpretations are available. The previously mentioned TCSEC, although traditionally used on stand-alone operating systems, has many concepts applicable to database systems. The Trusted Network Interpretation is a trusted computer/communications network systems interpretation of the TCSEC. Similarly, the Trusted DBMS Interpretation of the TCSEC, now under development, will cover stand-alone DBMS products.

TCB SUBSETS

Wouldn't it be of advantage to a vendor who ports a DBMS to many computers and to the evaluator not to have to evaluate the operating system of each target computer with the DBMS? If it can be shown that the DBMS does not interfere with the underlying security mechanisms of the os, then this can happen. The TCB or Trusted Computing Base is the totality of protection mechanisms in a computer system. The combination of these mechanisms is responsible for enforcing a security policy [DOD85]. A TCB Subset is a logical partition or layer of the TCB that enforces a subset of the security policies and supporting accountability policies enforced by the combined TCB [NCSC89]. With this approach, the TCB is divided into TCB Subsets, and each subset enforces a distinct part of the security policy

OTHER CONSIDERATIONS

A Trusted Path has been defined as a mechanism by which a person at a terminal can communicate directly with the TCB. To prevent spoofing, the mechanism cannot be imitated by untrusted software. A trusted path is also needed between the system security officer and the TCB.

In good software engineering, a design and development process that promotes modifiability, efficiency, reliability and understandability [BOOC83] should be used.

Finally appropriate audit mechanisms should be used. The issue is to get the granularity to record needed information while not severely impacting performance. To achieve this balance we have recommended the use of summary audit records to the TDI Chairman/Project Leader. Summary audit records log a count of the accesses for each subject accessing each level/compartment in a relation.

INFERENCE AND AGGREGATION

Inference and aggregation are big security problems. Inference is the derivation of information at a level for which the user is not permitted access by referencing other information to which he has access. In aggregation, the sensitivity level of a collection of data may be higher than the level of any individual datum. Therefore, in either case, the data's security label is not enough to protect the data. Neither is mentioned in the TCSEC. They are not specifically DBMS problems but are aggravated by the DBMS because the DBMS has been built to facilitate the manipulation and combination of data.

AN INFERENCE EXAMPLE

Who makes widgets? The answer is known but it is a secret. Is it company A, B, C, D or E?

It is known that widget makers need lots of water for cooling. Therefore the plant must be on a lake, river, etc. Also, they need lots of fossil fuel. Therefore the plant needs to be on a railroad siding or a barge pier. Finally, widget makers need chemical engineers.

The following additional information has been obtained from databases:
1. Company A is on a lake. Companies D and E are on rivers.
2. Companies A, C and E have railroad sidings.
3. Companies B and E advertise for Chemical Engineers.

Who? E.

INFERENCE/AGGREGATION CONTROLS

To control inference, and yet to keep classifications as low as possible, the applications designer, in a relational system, can classify table linkages or keys, but not the actual data in the tables. Or, the inference problems may be defined and the system could check queries for the problems. Control of aggregation could be done with query response history information. This however, presents a data aging/ system performance problem. That is, the more history you have, the better the control, but the longer it takes to scan the history.

SQL STANDARDS CONSIDERATIONS

"SQL" is a data definition and data manipulation language and is currently an ANSI standard. "SQL3", a proposed future ANSI standard, provides for triggers, mechanisms by which a user can affect the consistency of the database. Therefore the impact of SQL on integrity must be considered. Also SQL must be enriched to handle additions of audit, role and security level requirements.

CURRENT IMPLEMENTATIONS

Two types of implementations of current trusted database systems are the integrity lock and the Trusted Computing Base (TCB) implementations. Integrity lock methodology should be less costly but could involve covert channel problems [LAND88] and more direct attacks.

The integrity lock approach uses a trusted filter in front of an untrusted DBMS. The filter mediates all accesses between the users and the database, and performs trusted downgrades where necessary when providing at lower security levels with data from the database. [WINK89] A trusted operating system at least the filter level and B1 or higher is required to enforce the separation between DBMS end users. Both discretionary and mandatory access controls are at least in part located in the filter.

The TCB implementations place the assurance and security functionality in a relatively small kernel of code. The smallness of the kernel invites verification and other proofs of correctness. The TCB may be broken into subsets, with each subset enforcing a part of the policy.

NCSC DISCRETIONARY SECURITY PROTOTYPE CONSIDERATIONS

Some of the factors considered in the "C2" prototypes developed at the NCSC are:

- discretionary access control
- object reuse
- identification and authentication
- audit
- security testing
- data integrity
- performance

NCSC MANDATORY SECURITY PROTOTYPE CONSIDERATIONS

In addition to the "C2" prototype considerations, the following are being considered in the "B"-level prototypes developed at NCSC:

- labels
- label integrity
- exportation to
    - multilevel hosts
    - single level hosts
- exportation of labeled information
- mandatory access control

DISTRIBUTED DATABASE MANAGEMENT SYSTEMS (DDBMS)

Distributed database management systems form an important set of security problems and opportunities. This type of DBMS has multiple sites connected together into a communications network in which a user at any site can access data at any site. Characteristics of this DBMS may include the physical location of the data being transparent to the user, redundant data for performance and heterogeneous nodes. Vendors who have current implementations include CCA, Oracle and Ingres. Maintaining the consistency of replicated copies of data may be a problem.

The DDBMS may be very efficient because data can be stored where the user uses it. Data can be better controlled by isolating it on particular nodes. The DDBMS, with multiple nodes and redundant data and communication paths answers the system availability or denial of service problem. System performance may be enhanced by local storage of frequent used data and by other distribution of data. Also, there are opportunities for the parallel execution of queries.

Problems also are many. How do you maintain database consistency with redundant data during updates/deletes and restores? What is the best method of identification and authentication? What is the best way to audit? Deadlocks must be controlled and priorities maintained. Other problems include the construction of a distributed MTCB, the part of the TCB that manages mandatory access control. Also, we must look at the distributed management of DAC, the Discretionary Access Control, and the problem of the consistency of DAC on replicated tables. How do you handle distributed transactions? Can serializability be maintained without creating inference channels? Can we use weak consistency? Are there new covert channels? A subsetted TCB could be very large and complex and therefore difficult to verify.

Encryption would be very useful between nodes and to store data. Long term keys are a problem. What algorithms should be used? How does this affect performance? How should the DDBMS be administered? What tools are needed? How do you resolve heterogeneous security policies? How do you assure the security of the system?

SUMMARY

Database security is a young interdisciplinary science, filled with promise and opportunities. The demand already exists. C-level operating systems are here and b-level operating systems are appearing. The evaluators tool, the Trusted Database Interpretations, is being written. Trusted DBMS prototypes are being produced. These include LOCK Dataviews, Trusted Oracle, Secure Data Views (SeaView), and Teradata. Sybase and Trudata have products. New products are emerging. In the future there will be an increasing demand for database security. Many databases will be very large, distributed and with heterogeneous nodes. Databases will be smart, with multimedia data, where rules, and derived knowledge are stored and used. Parallel, array and fault tolerant processing will be the norm. Operating systems may have some database management system functionality. Security research and development is needed in all of these areas.

GLOSSARY

aggregation problem - The aggregation problem refers to the fact that the sensitivity level of a collection of data may exceed the sensitivity level of any individual datum in that collection. [NCSC89]

B - A TCSEC Division. The notion of a TCB that preserves the integrity of sensitivity labels and uses them to enforce a set of mandatory access control rules is a major requirement in this division. Systems in this division must carry the sensitivity labels with major data structures in the system. [DOD85]

C2 - A TCSEC class. Systems in this class enforce a more finely grained discretionary access control than C1 systems, making users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation. [DOD85]

Discretionary Access Control - A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). [DOD85]

inference - derivation of new information from known information. The inference problem refers to the fact that the derived information may be classified at a level for which the user is not cleared. [NCSC89]

Mandatory Access Control - A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. [DOD85]

REFERENCES:

BELL73 Bell, D., and L. Lapadula, **"Secure Computer Systems: Mathematical Foundations and Model"**, MITRE Report MTR 2547, v2 Nov 1973.

BIBA77 Biba, K., **"Integrity Considerations for Secure Computer Systems"**, U.S. Air Force Electronic Systems Division, 1977.

BOOC83 Booch, Grady, **Software engineering with Ada**, Menlo Park: the Benjamin Cummings Publishing Company, 1983.

DATE86 Date, C. J., **An Introduction to Database Systems**, Reading, MA: Addison-Wesley, 1986.

DENN88 Denning, D. E., **"Lessons Learned From Modeling a Secure Multilevel Relational Database System"**, Database Security: Status and Prospects, Amsterdam: Elsevier Science Publishers, 1988.

DOD85 DoD, **Department of Defense Trusted Computer System Evaluation Criteria**, DOD 5200.28-STD, 1985.

JACO89 **"Security In Computing"**, Computer, August, 1989, p. 150.

LAND88 Landwehr, C. E., **"Database Security, Where Are We?"**, Database Security: Status and Prospects, Amsterdam, Elsevier Science publishers, 1988.

NCSC88a National Computer Security Center, **Glossary of Computer Security Terms**, NCSC-TG-004-88, 1988.

NCS288b National Computer Security Center, **Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria**, NCSC-TG-005, 1987.

NCSC89 National Computer Security Center, **Draft Trusted DBMS Interpretation of the DoD Trusted Computer System Evaluation Criteria**, 1989.

WINK89 Winkler-Parenty, H., **"Can You Trust Your DBMS"**, Database Programming & Design, July 1989, pp. 50-59.

Executive Summary

1990: A YEAR OF PROGRESS IN TRUSTED DATABASE SYSTEMS

John R. Campbell
National Computer Security Center
Office of Research and Development
9800 Savage Road
Fort George G. Meade, Maryland 20755-6000
(301) 859-4488

1990 has been a year of progress in trusted database systems. It is the year when research and development has reached new level of maturity. Tougher questions, such as inference, aggregation and concurrency control are being examined, and answered. Theoretical foundations are being established. At the Third RADC Workshop on Multilevel Security, Bhavani Thurasingham of MITRE proved that the general inference problem is unsolvable and then presented a series of approaches that could be used to control inference. Knowing that the general problem is unsolvable enables us to more efficiently apply resources to research the controlling methodology.

At the same Workshop, TY Lin of California State University provided insights into aggregation by looking at aggregation through theoretical constructs. Several researchers, each of whom by their work have developed perspectives into the problem of polyinstantiation, discussed and debated this problem. The researchers who attended this Workshop, and their work, were of high quality.

The activity and productivity in trusted database systems was also seen at the 1990 IEEE Computer Society Symposium in Security and Privacy. Three of the eleven sessions discussed work being done in this area.

1990 is also the year of the trusted multilevel database prototype. At the NCSC, Oracle Corp. delivered a "B-level" prototype in April. This prototype is exploring research questions, methodologies and techniques in building multilevel trusted systems. The papers of Vetter, on TCB-Subsets and Maimone on concurrency controls (Tuscon, winter, 1990) are outputs of this work. Oracle is delivering a second prototype, based on the Gemsos Operating System, in August.

Teradata Corp. is building a trusted "B-level" database machine for the NCSC. This computer, dedicated to database management system processing, can be accessed by several host computers. It has an interesting MIMD architecture, is modular and fault tolerant. Systems can be configured from six to several hundred processors. Discussions on metadata host control were interesting.

Sybase Corp. has produced a "B-level" server. In doing this, much work on trusted subjects and covert channels was done.

In this paper, I have enclosed all trust ratings in quotes. This is done because no systems have been evaluated by the NCSC. No system has been evaluated because the "yardstick" for evaluations has not yet been built. However, it is likely that this yardstick, the first edition of the "Trusted Database Interpretations of the Trusted Computer System Evaluation Criteria", or "TDI", will be published before the Conference. This is the document that will be used to evaluate the security of database systems.

1990 is a year for users too. Users have, and will soon have still more security in off-the-shelf database systems. For example, "C2-level" security will be part of the standard database package from both Oracle and Teradata. Sybase, Trudata and others have packages.

1990 hopefully is also a good year for future plans for research and development in trusted database systems. The NCSC is sponsoring two parallel efforts to develop highly secure database systems. "Secure" here means secrecy, integrity and availability. Secrecy will be at the "B-3" and "A-1" levels. "Integrity" includes system, label and data integrity. "Data integrity" includes entity, referential and other integrity. "Availability" includes fault tolerance and distributed systems. The efforts will last from five to six years. One effort uses host-based architecture; the second uses the database machine.

To date much has been accomplished, much is being done. If this area is properly supported, it is likely that future accomplishments will be equally bright.

Executive Summary

Secure Database Products

James Pierce
Teradata Corporation

DBC/1012 Security Features

C2 General Release - June, 1990

- DAC
- Auditing
- Logon Control

B1 Prototype - February, 1991

- MAC
- Audit Extensions
- Password Guidelines

Implementation Experience

Database Computer

- Parallel Architecture
- Dedicated Operating System
- Identification & Authentication

Audit in Large Databases

- Performance
- Storage Requirements

Labels of Metadata

- Requires Trusted Process

Executive Summary

Trusted Database Software: Review and Future Directions

Peter J. Sell

Office of Research and Development
National Computer Security Center

The Trusted Database Software (TDS) program is a two phase effort using existing technologies and commercially available Database Management Systems (DBMS) to develop a series of trusted DBMS prototypes. These prototypes range from a stand-alone C2 level prototype to a distributed A1 level prototype. The first goal of this project is to develop secure DBMS prototypes, both distributed and non-distributed, which will provide the Government with secure database management capabilities. The second goal of this project is to have the prototypes developed into commercial products, evaluated by the National Computer Security Center (NCSC), and added to the Evaluated Products List (EPL).

Phase One

The first phase of the TDS project involves the development of five prototypes. The first two prototypes implement discretionary access control (DAC) and run on the VAX VMS operating system and the Gemini Secure Operating System (GEMSOS). The next three prototypes implement mandatory access control (MAC) and run on the VAX VMS, the VAX Security Enhanced VMS (SEVMS), and the GEMSOS operating systems. In July 1988, Oracle Corporation signed a contract to develop the prototypes for this phase of the TDS program.

Oracle Corp. delivered and installed the first prototype developed under this contract to the NCSC in May 1989. This prototype, which runs on the VMS operating system, implements DAC at a potential C2 level of trust as defined in the Trusted Computer System Evaluation Criteria (TCSEC). Besides the TCSEC security requirements, the prototypes also include two research topics, referential integrity and group access controls. Referential integrity insures that all foreign keys refer to legitimate values defined in a different table. Group access controls allow database administrators to separate users into groups and then assign privileges to those groups. This prototype provides group access controls by using roles, or bundles of privileges. For example, a role named payroll could be created that contained all the privileges relating to a person working in the payroll department. A database administrator, or another privileged user, may grant this role to other users as if it were a single privilege.

Oracle Corp. delivered and installed the second prototype, which runs on the VMS operating system, at the NCSC in April 1990. This prototype was a stepping stone to the third prototype that runs on the SEVMS operating system. Oracle delivered and installed this prototype at the NCSC in May 1990. This prototype enforces MAC at a potential B1 level of trust as defined in the TCSEC. As with the first prototype, this prototype includes referential integrity and group access controls.

The final two prototypes will be delivered in the fall of 1990. The first of these prototypes implements DAC and the second of these prototype implements MAC. Both prototypes run on the GEMSOS operating system.

Phase Two

The second phase of the TDS program will produce four prototypes at the potential B3 and A1 levels of the TCSEC. The first two prototypes will operate in a stand-alone environment, while the second two prototypes will operate in a distributed environment. This project incorporates the requirements of the TCSEC, the Trusted Network Interpretations of the TCSEC (TNI), and the Trusted Database Interpretations of the TCSEC (TDI). These prototypes incorporate all three of these requirements on one system for the first time.

The first prototype will run in a stand-alone environment. This prototype will meet all the requirements of a potential A1 system except that the prototype will not meet the formal verification requirements of the TCSEC. The second prototype will then undergo formal verification to meet a potential A1 level of trust. The third prototype will run in a three-node distributed environment and will meet all the requirements of a potential A1 distributed system except the formal verification requirements. The fourth, and final, prototype will then undergo formal verification to meet a potential A1 level of trust for a distributed system.

In addition to the TCSEC requirements, several security research topics will be studied. The first topic to be studied is a data integrity policy combined with a security policy. All four of the prototypes developed for this phase of the program will enforce a combined data integrity and security policy. The second research topic to be studied is polyinstantiation. Polyinstantiation allows multiple copies of a record to be stored at different classifications. Inference and aggregation controls will be studied as the third research topic and an inference and aggregation control policy will be included on the prototypes. Finally, the prototypes will maximize the use of Trusted Computing Base (TCB) subsets in order to increase the portability and simplify the evaluation of the prototypes.

Several non-security requirements also will be included on the prototypes. The first requirement is to maximize the performance of the DBMS by designing security into the system from the beginning. The second requirement is that the prototypes implement ANSI SQL as the query language. Any enhancements to the SQL language, as a result of the addition of multi-level security, will remain consistent with ANSI SQL.

The Center sent a Request for Proposal (RFP) to several vendors in the DBMS arena in May 1990 for this phase of the TDS program. A contract should be signed by the end of 1990.

Trusted Systems Interoperability

Helena B. Winkler-Parenty

Sybase, Inc.
6475 Christie Avenue
Emeryville, CA 94608

Past Year's Efforts

During the past year, Sybase has continued its commitment to producing trusted DBMS technology. The first offering of trusted products was the SYBASE Secure SQL Server^{TM} and SYBASE Secure SQL Toolset^{TM}. These products were generally available at the end of 1989. Since then, SYBASE Open Server^{TM}, an application programming interface for developing server applications, has been extended to incorporate secure features. In addition, the Secure SQL Toolset has been ported to several platforms, including Digital's SE/VMS and Sun OS MLS.

This past year there have been several major trusted projects ongoing within Sybase. After shipping Release 1 of the Secure SQL Server we have been providing customer support for the product, and shipped a maintenance release in the middle of the year. We gathered information from our customers to feed into the design process of Release 2. Release 2 has been fully specified and designed, and implementation is now underway.

We used this past year as an opportunity to reexamine our B2 strategy. When we initially started work on our B2 targeted DBMS there were no trusted operating systems that could be used as a base. There are now several trusted operating systems in the marketplace, and others that are being developed.

The final project that we have worked on over the last year has been porting the Secure SQL Toolset to SE/VMS and Sun OS MLS. It has been an interesting learning experience seeing how each of these multi-level operating systems treats trusted processes both running on top of themselves and on top of other operating systems. Heterogeneous trusted systems working cooperatively is a hard but important problem. Creating a common label space and enabling our Secure SQL Server to communicate with users at multiple security levels were two of the challenges we faced.

SE/VMS

One of the two MLS operating systems we targeted for the Secure SQL Toolset was SE/VMS. For both security and user convenience we decided to modify the Toolset to retrieve the user's operating system login security level from SE/VMS, and use this security level as part of the identification and authentication procedure with the Secure SQL Server. This means that users do not need to retype their login security level when logging into the Secure SQL Server.

In addition, it is important for site security, because if the user logs into the Secure SQL Server at a different level from their operating system session level, security breaches can occur. If the user's operating system level is greater than their DBMS level, then trojan horse software or an error on the user's part can cause data at a high security level to be inserted into the database and labeled at a lower security level. Similarly, if the user logs into the DBMS at a higher level, high level data could be retrieved from the DBMS and be incorrectly inserted into a lower level operating system file. Both of these potential problems are averted by the Sybase trusted system.

Once the user's login security level has been retrieved from SE/VMS, the Secure SQL Toolset converts it from the format used by SE/VMS to the format used by Sybase. SE/VMS supports 256 security levels and 128 compartments, if integrity labels are not used. Release 1 of the Secure SQL Server supports 16 hierarchical levels and 64 compartments. Release 2 of the Secure SQL Server will support 256 levels and 1024 compartments. If the user has logged into SE/VMS with a level between 16 and 256, or compartments 65 through 128, and they are attempting to login to a Release 1 Secure SQL Server then their login will be rejected. In addition, the Secure SQL Server confirms that the user's maximum allowable DBMS login level dominates the user's active SE/VMS session level. Otherwise, the login request to the Secure SQL Server is denied.

In SE/VMS it is both possible and easy to specify that another machine, and all of the software running on it, is trusted to operate over a specified range of security levels. Therefore, if the Secure SQL Toolset is running on SE/VMS and the Secure SQL Server is on another operating system, the only special action that needs to occur is for someone with root privileges to specify that the machine which is running the Secure SQL Server is trusted to operate over the range of at least level 0 with no compartments on, to level 15 with compartments 1 through 64 on. This will need to be done for each SE/VMS machine running the Secure SQL Toolset that wants to connect to the Secure SQL Server.

Sun OS MLS

For the Secure SQL Toolset to run on Sun OS MLS similar changes needed to be made to retrieve the user's Sun OS MLS login level, convert it to the Sybase format, and send it to the Secure SQL Server.

A significant difference between SE/VMS and Sun OS MLS is that Sun OS MLS will only consider another machine to be trusted if it complies with the change to the IP header that Sun OS MLS has specified. Sun OS MLS requires that the Internet Protocol (IP) header contain the security level of the packet. Since the Secure SQL Server is an application running on top of an operating system Sybase could not make this change. In order for the Secure SQL Server to communicate with users at multiple levels we were required to write a new utility called the Trusted Multiplexor.

The Trusted Multiplexor is trusted by Sun OS MLS to operate over a range of security levels, and is therefore part of the Trusted Computing Base (TCB). It runs on each client machine, receiving requests from clients at their login level, writing the request up to Sun OS MLS system high, and then passing these requests over the network to the Secure SQL Server. After the Secure SQL Server has formulated a response to the user's query, the Trusted Multiplexor receives the response from the Secure SQL Server.

writes it to the user's login security level, and sends the response to the user.

All communication with the Secure SQL Server is over the network and is labeled at Sun OS MLS system high, regardless of the actual level of the data being transmitted. Because data is overclassified, and is never underclassified, no security breaches can occur.

After installing the Secure SQL Server a user with root privileges must specify on every client machine that may want to access the Secure SQL Server that the machine running the Secure SQL Server is a single level machine running at Sun OS MLS system high. In addition, this same user must install the Trusted Multiplexor. The Trusted Multiplexor is installed at system low, with root privileges. These privileges allow the Trusted Multiplexor to communicate with users at all possible security levels.

Conclusion

1990 was a busy year for Sybase. We've received favorable feedback on our secure products, which have been generally available for a year. We've ported the Secure SQL Toolset to two MLS operating systems, SE/VMS and Sun OS MLS. We've addressed and solved installation, site security, and interoperability problems with running our trusted products in a heterogeneous trusted operating system environment.

Oracle Secure Systems 1989-1990 A Year In Review

Linda L. Vetter
Director, Secure Systems
Oracle Corporation

Over the last year, Oracle Corporation has continued to focus significant efforts on trusted relational database management system (RDBMS) technology through our on-going research, development, and consulting activities. Oracle Secure Systems, formed in February 1989 as an Oracle Corporation strategic business unit, is chartered with spearheading Oracle's efforts to research, design, build and deliver high security RDBMS commercial off-the-shelf software to commercial and government organizations worldwide.

At the Computer Security Conference last October, I discussed Oracle's discretionary prototype that was delivered to the NCSC in May 1989 running on Digital Equipment Corporation's Class C2 VAX/VMS trusted computing base (TCB). That prototype was the first in a series of working "proof of concept" prototypes of trusted RDBMSs that run on various secure platforms and which are targeted for Trusted Computer Security Evaluation Criteria (Orange Book) evaluation Classes C2 and B1 under an R&D contract awarded to Oracle Corporation in July 1988.

In the twelve months from October 1989 to October 1990, Oracle has achieved several additional milestones, including the delivery of our B1 level RDBMS prototype to the National Computer Security Center (NCSC). Building on last year's C2-style prototype, Oracle Corporation developed a mandatory prototype that supports multilevel security (MLS) features including data labeling and other Class B1 functionality. The ORACLE MLS mandatory prototype was delivered to the NCSC in April 1990 running on DEC's target Class B1 operating system, SEVMS.

The security policy of the ORACLE mandatory prototype consists of two types of security: mandatory and discretionary. Mandatory security controls access to information through labels: in order to access certain information, a subject's label must meet specified criteria in relation to the label of requested objects. Discretionary security further controls access to information through privileges: in order to access certain information a subject must possess the appropriate privilege as granted by the object's owner or security officer.

The ORACLE mandatory prototype implements mandatory and discretionary security through a constrained TCB subset architecture in which the RDBMS builds upon security features of the trusted computing base (TCB) to which it is ported. This architectural approach extends security features already provided by a trusted operating system with complementary security capabilities in the RDBMS. Together, the RDBMS and operating system (OS) jointly and cooperatively enforce the stated security policy.

The constrained TCB subset architecture requires a clear definition between the RDBMS portion of the TCB and the operating system and hardware portions of the TCB on which the RDBMS runs. Because the multilevel secure (MLS) ORACLE prototype's implementation of a constrained TCB subset architecture relies on mandatory security mechanisms in the OS base, portions of the MLS ORACLE security policy are based on the operating system mandatory security policy. The RDBMS implements database-relevant discretionary security without superseding or violating the security policy of the underlying secure operating system.

In the mandatory prototype, the operating system mandatory TCB (m-TCB) controls all access to storage objects, for example OS files and memory segments. The m-TCB subjects execute code of appropriately labeled ORACLE instances. Database subjects map directly to OS subjects that have successfully completed identification and authentication by the operating system m-TCB identification and authentication component. Discretionary access control on database objects, such as views and tables, and database auditing are enforced by the RDBMS prototype.

Oracle's implementation of the constrained TCB subset approach takes advantage of the fact that our MLS RDBMS is designed to be portable to any B1+ -targeted platform and that it can depend on certain features, functions, and assurances already available in such environments. For example, MLS ORACLE uses the user ids and passwords, user clearance ranges, label definitions, MAC policy model (of label dominance), and other security facilities of the underlying B1+ platform. Not only does this reduce the amount of code within the RDBMS that must be trusted, evaluated, and RAMPed, but this also means that users only need to maintain and change their passwords in only one place; also, installation security officers do not need to define and maintain clearance ranges, valid label formats, etc., more than once. MLS ORACLE provides installation options allowing the database-specific audit trail records to be maintained inside the trusted DBMS or to be delivered to the secure OS for global recording.

Oracle Secure Systems is now completing the final phase of its NCSC contract. The final phase consists of the ORACLE prototypes ported to Gemini Computers' target Class A1 GEMSOS TCB hardware and software. Gemini Computers is a subcontractor to Oracle Corporation on this R&D contract.

The ORACLE C2 and B1 style prototypes delivered to the NCSC included extensive test suites and documentation deliverables similar to those required for NCSC evaluations, plus training courses on porting, installing, and using multilevel secure systems. Features of both the mandatory and discretionary prototypes, including enhanced documentation and training materials in addition to product enhancements, are being incorporated into ORACLE RDBMS commercial products and services such as Trusted ORACLE RDBMS, which is targeted for B1 and higher ratings on multiple secure platforms.

Another major R&D effort involving Oracle Secure Systems during this year has been the SEcure DAta Views (SeaView) high security RDBMS project sponsored by the United States Air Force's Rome Air Development Center (RADC). In this R&D contract awarded at the end of 1989 as a follow-on to the original RADC SeaView project, Oracle Corporation is teamed with SRI International (prime) and Gemini Computers.

The original SeaView project was a three-year study completed by SRI in January 1989. That effort produced a security policy, a model, an implementation specification, and a GEMSOS-based demonstration system for a multilevel secure (MLS) RDBMS designed to meet high United States Department of Defense standards for computer security. ORACLE RDBMS was selected as the commercial database that best met the original SeaView project goals. These goals included:

- Portability to multiple high security operating systems
- Full function relational or structured query language (SQL) interface
- Support of user-defined views and discretionary access controls for both base relations and views

In the follow-on effort, Oracle Secure Systems is participating in the production of a demonstration system of a Class A1 secure RDBMS based on the original SeaView design.

The SeaView design supports multilevel relations, a technology that provides classification and access control of individual database elements based on "need-to-know" criteria. SeaView's hierarchical subset architecture uses an ORACLE RDBMS MLS prototype, extended from work under the NCSC prototype contract, layered between Gemini's A1-targeted GEMSOS TCB and the "untrusted" SRI/SeaView MSQL (multilevel SQL) Processor. When completed, the SeaView demonstration system will further allow users to define, manipulate, and control relational database entities and relations at a very fine level of granularity in a multilevel, high assurance environment.

In building these secure systems, we have attempted to ensure that the required mandatory and discretionary security enforcement is applied in the RDBMS and OS base products so that our wide range of application development tools and utilities, as well as existing applications, can all continue to process as untrusted systems with minimal or no modification. Besides reducing possible impact on developers and system maintenance personnel, this greatly simplifies the efforts of system evaluators, certifiers and accreditors.

Obviously, it has been a busy twelve months for Oracle Secure Systems. Participation in these and other secure database activities are helping Oracle Corporation build an integrated family of portable, high-performance, commercially available secure relational database products for a wide variety of platforms.

Executive Summary

Trusted Database Machine Program: An Overview

William O. Wesley, Jr.

Office of Research and Development
National Computer Security Center

Program Goal and Description

The goal of the Trusted Database Machine (TDM) program is to develop a solution to the database security problem of providing multilevel security for database machines. The overall program will be broken into 4 phases. Each phase will focus on the development of demonstratable secure database machine prototypes, using existing technology and a commercially available database machine, making the commercially available database machine a more secure product.

The first phase will focus on the development of 2 secure prototypes. The first prototype, in phase 1, will incorporate "C2" security features into a database machine. The second prototype, in phase 1, will incorporate "B1" security features into a database machine. A detailed description of what "C2" and "B1" security requirements are may be found in the Department of Defense Trusted Computer System Evaluation Criteria, document number DOD 5200.28-STD.

The second phase of the program will focus on the development of 1 secure prototype. This prototype will require the development of a "A1" database machine. Unlike phase 1, security features at the "A1" level cannot be incorporated into an existing database machine. "A1" security features must be a part of the entire development process of the database machine. Further information regarding "A1" security features may be found in the Department of Defense Trusted Computer System Evaluation Criteria, document number DOD 5200.28-STD. In addition to addressing the TCSEC requirement for "A1", phase 2 will also address several database security issues. These issues include - the incorporation of a security policy and a integrity policy and how they will work together, inference and aggregation, data integrity, referential integrity, and system performance and how the security controls affect it.

The third phase of the TDM Program will focus on the development of an "A1" networked database machine. This portion of the program will incorporate the results from phase 2 of the program and develop a secure network at the "A1" level, which will allow several heterogeneous computers (both mainframes and workstations) to use the secure database machines resources in a secure manner. Further information regrading "A1" secure networks may be found in the National Computer Security Center Trusted Network Interpretation, document number NCSC-TG-005.

The fourth and final phase of the program will develop an "A1" secure heterogenous multi-node database machine system. This phase will securely link together several database machine systems with several heterogenous computers (both mainframes and workstations), providing a secure distributed operating environment for the entire system.

Accomplishments to Date

Phase 1

The contract for the phase 1 was awarded to the Teradata Corporation in September 1987. In August 1988, Teradata completed the development of the "C2" prototype. This prototype implements such features as discretionary access control, object reuse, identification and authentication, and auditing of named objects. To date, Teradata is working on the "B1" secure prototype development. The "B1" prototype is scheduled to be completed in FY91. In addition to the security features implemented in the "C2" prototype, the "B1" prototype will also implement manadatory access control, and labeling. Teradata has recently incorporated the security features implemented in the "C2" prototype, into their commercial version 4.2.

Phase 2

The RFP for the phase 2 work has been developed and released. The RFP is a competitive bid and a contract is expected to be awarded by the end of the calendar year.

Executive Summary

TRUSTED DATABASE SYSTEMS: THE TOUGH ISSUES

John R. Campbell
National Computer Security Center
Office of Research and Development
9800 Savage Road
Fort George G. Meade, Maryland 20755-6000
(301) 859-4488

Research and development in trusted database systems have reached a new level of maturity. The "B-level" prototypes being produced attest to this maturity. However, many interesting problems remain.

A. CONCURRENCY CONTROLS

For example, what is the right type of concurrency control for trusted multilevel database systems? Database systems frequently have many users working on the system at the same time. Each is doing a part of their own transaction. It is up to the system to insure that each user's transaction is consistent.

Suppose A and B are tellers that are subtracting cancelled checks from checking accounts. Bob has written two checks, one for $10, one for $20. Bob's checking account currently has $100 in it. The processing is done in this sequence. Clerk A reads Bob's balance, $100. Clerk B read's Bob's balance, $100. Clerk A subtracts the amount of the first check from the balance, $100 - $10, and writes the difference, $90, in the database. Clerk B subtracts the amount of the second check from the balance he read, $100 - $20, and writes the difference in Bob's account, or $80. Bob has spent $20 plus $10, or $30; he is charged for $20. This is a concurrency problem.

One solution is for the system, or clerk A through the system, to lock all other users out of Bob's balance until clerk A is finished updating the balance. The disadvantage of this technique is that it impacts performance, especially if the structure locked is large.

Multiple users on a multilevel system who are able to lock the same data create another problem, that of a covert channel. Locking can be used to signal high to low.

The issue then is to provide concurrency with minimum impact on performance and with covert channels removed, if possible.

B. LABELS

A sensitivity label represents the security level of an object. It describes the classification of the data in the object. What is the "right" way to label objects? Do we need separate labels for secrecy, including compartments, integrity and other designations? How do we handle special needs, for example, 1024 compartments? Should we, and how do we standardize labels?

C. AUDIT

An audit trail provides documentary evidence of processing. It associates users with objects. How do you do multilevel audit in a trusted database management system? Do you have an audit trail at each level? How are these combined?

What should the granularity of the audit trail be and what should be recorded? If the granularity is too fine or if every act is recorded a huge amount of audit data is created. On the other hand, insufficient audit data will hinder and even prevent the detection or recording of an inappropriate event.

What assurance is needed? The processes used to create the audit trail and the audit trail itself should be tamperproof. How do you do this?

How is the operating system involved? Certainly there are situations where the database system is incapable of creating its own trail; for example, when it is coming up or crashing. Another example is the deletion of the audit trail when the disks are full; this is another case where the operating system audit trail may be needed. What if the operating system is used for identification and authentication?

How do you combine audit trails? In the distributed case, how should the separate streams be combined? Do systems distributed across the country use standard time stamps?

What tools are necessary so that system security officers may obtain meaningful information out of the mass of audit data that was created?

D. HIGH ASSURANCE

"B3" and "A1" systems require high assurance. How do you get this assurance? Database systems are large and their TCB's are likely also to be large. Are the current verification tools adequate for the job?

Must the SQL compiler be in the TCB? If it does, the TCB will be very large. What software engineering methodologies and other techniques can be used to provide the SQL compiler with adequate assurance?

E. IMPLEMENTING DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

How do you implement a trusted multilevel distributed database management system? Do you use client-server, distributed operating system or distributed database system? Which is easiest?

F. INTEGRITY

How do you build integrity into a trusted multilevel database system? What is "integrity"? How do you resolve the conflict between integrity and secrecy, for example, referential integrity and secrecy?

G. TRUSTED SUBJECTS

Trusted subjects are subjects that violate the security policy of the system and are trusted to do that violation in a very specific way. A downgrader, for example, is an example of a trusted subject. Some trusted multilevel database systems are implemented using trusted subjects. Should we be doing this? How do trusted subjects affect evaluations?

H. TCB SUBSETS

TCB subsets are layers of the TCB. Each enforces a specific part of the overall security policy. If an upper layer does not interfere with a lower layer, then it may not be necessary for the lower layer to be reevaluated. For example, if the lower level, an operating system, provides mandatory access control (MAC), the upper, a database system, provides discretionary access control, and the upper level does not interfere with MAC enforcement, then the operating system does not have to be reevaluated when evaluating the upper layer. this is a big labor savings. Is this the way to go?

I. COVERT CHANNELS

Covert channels are communications channels that allows processes to transfer information in a way that violates the security policy of the system. Shared resources invite covert channels. Database systems make a point out of sharing resources. Where are the covert channels? How do you eliminate or control them?

This is but a partial list of tough database security problems. These problems are currently being studied, and, one by one being solved. Highly trusted multilevel database systems will be a reality.

TOUGH ISSUES:
INTEGRITY AND AUDITING IN MULTILEVEL SECURE DATABASES
(Panel Position Paper)

Sushil Jajodia

Department of Information Systems and Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030-4444

I. INTEGRITY

The requirements for maintenance of database integrity are often in conflict with the security[†] requirements of multilevel secure database management systems. Nevertheless, although we may never reach a point at which security and integrity feel entirely comfortable with each other, it is possible to have them live in harmony to a large extent.

Security requires that users be denied all knowledge of data to which they do not have authorized access. On the other hand, integrity constraints can be defined over data in different access classes in such a way that information from one access class can leak to another. This conflict is exacerbated by the well-known covert channel problem: the fact that a leak between access classes can provide, not only information about data over which the constraint is defined, but signaling channels by which information about data at one access class can be passed to users at another access class. If satisfaction of an integrity constraint requires that changes to data at one access class be reflected indirectly in the value of data at another class, then a Trojan Horse program embedded in a process at the first class can encode data at the first class as data at the second class by varying the data involved in the integrity constraint so that it produces detectable changes in the data in the second class. Such a channel could be used to pass on, not only information directly affected by the constraint, but any other information to which the Trojan Horse has access. Thus integrity constraints that seem intuitively harmless can actually be more dangerous than they appear at first glance.

The requirements for database security can be summed up in a general statement, called the Basic Security Principle for Multilevel Secure Databases [1,2]. (One access class is said to *dominate* another if a user who is cleared to the first access class is also assumed to be cleared to the second access class.)

[†] Perhaps I should use 'secrecy' instead of 'security' throughout.

Basic Security Principle for Multilevel Secure Databases
The access class of a datum should dominate the access classes of all data affecting it.

The reason for the Basic Principle is clear: if the value of a datum can be affected by data at access classes not dominated by its own, information can flow into the item from those other access classes.

We will find it useful to divide the definition of integrity into three different (although somewhat overlapping) properties:

Correctness
: The correctness of a database is the degree to which it satisfies all known consistency constraints.

Availability
: The availability of a database is the degree to which data can be made available to authorized users.

Unambiguity
: The unambiguity of a database is the degree to which queries have identical responses. We take unambiguity to apply to equivalent as well as identical queries. In other words, if two equivalent queries have different responses, we consider the responses to be ambiguous.

A fourth property is also important to the maintenance of database integrity.

Recoverability
: The recoverability of a database is the degree to which, whenever it is not correct, available, or unambiguous, it can be returned to such a state.

If we examine the various kinds of threats to integrity that arise in multilevel secure database systems, there are several ways that exist of dealing with them. (See [1,2].) Whenever a conflict arises between database integrity and security, it can often be resolved more or less satisfactorily by introducing a trade-off among the three integrity properties instead of between integrity and security. In general, of all three database integrity properties, unambiguity is the most easy to sacrifice, perhaps because unambiguity is the most easily recoverable of the three properties. When data availability is sacrificed, data may become irretrievably lost, and when data correctness is sacrificed, data may become irretrievably corrupted. However, it is always possible to recover database unambiguity as long as well-defined rules exist for identifying the correct version of the data. The definition of such rules is nontrivial and remains an open problem.

II. AUDITING

In the Trusted Computer System Evaluation Criteria [3], the accountability control objective is stated to be as follows:

> ''Systems that are used to process or handle classified or sensitive information must assure individual accountability whenever either a mandatory or discretionary security policy is invoked. Furthermore, to assure accountability the capability must exist for an authorized and competent agent to access and evaluate accountability information by a secure means, within a reasonable amount of time and without undue difficulty.''

Existing databases clearly fail to meet the above objective. In my view, this is because existing database systems treat only the current data in a systematic manner; the old information is either spooled to the log that has an *ad hoc* structure or is deleted. If we wish to meet above requirements in a database, we need to treat the audit data in a systematic way as well, not just the current data. Once this is done, we can then begin to address other issues related to audit. (See [4].)

A similar situation existed in the early 1960's. An application program used its own specially designed files. As a consequence, it was very difficult for a user to know what files already existed in the system. Knowing the existence of a file was not sufficient; the user needed to know the actual file structure as well. If a file maintained by some program was reorganized, there was no assurance that other programs wishing to access the reorganized file would still work. Thus, much information was stored redundantly; however, this there were problems when users started to look for consistency of results.

The database approach in the early 1970's overcame many of the problems listed in the previous paragraph. An important tool called a *data model* was developed which imposed a logical structure on all the (current) data in the system. This allowed the users to see data, not as an arbitrary collection of files, but in more understandable terms. Database researchers developed another key concept of *independence*: The logical structure of the data became independent of the details of physical storage of data. Since now users could reorganize the physical scheme without changing the logic of existing programs, duplication of data could be avoided.

We must take a similar approach when it comes to audit data. First, we must carefully list the audit objectives, and then provide mechanisms required to achieve these objectives. It is my position that in an audit trail, it should be possible to audit every event in a database. I call it ''zero-information loss.'' What events are actually audited depends on the sensitivity of the events in question and the results of a careful risk analysis. In addition to the actual recording of all events that take place in the database, a logical structure needs to be imposed on audit data. An audit trail requires mechanisms for a complete reconstruction of every action taken against the database: *who* has been accessing *what* data, *when,* and in what *order*. Finally, an audit trail must also provide the capability (a query language) to easily access and tools to evaluate the audit information.

References

1. Catherine Meadows and Sushil Jajodia, "Integrity versus security in multi-level secure databases," in *Database Security: Status and Prospects*, ed. Carl E. Landwehr, pp. 89-101, North-Holland, Amsterdam, 1988.

2. Catherine Meadows and Sushil Jajodia, "Maintaining correctness, availability, and unambiguity in trusted database management systems," *Proc. 4th Aerospace Computer Security Applications Conference*, pp. 106-110, December 1988.

3. "Department of Defense Trusted Computer System Evaluation Criteria," Department of Defense, National Computer Security Center, December 1985.

4. Sushil Jajodia, Shashi K. Gadia, Gautam Bhargava, and Edgar H. Sibley, "Audit trail organization in relational databases," in *Database Security, III: Status and Prospects*, ed. D. L. Spooner and C. Landwehr, pp. 269-281, North-Holland, Amsterdam, 1990.

Issues of Concurrency Control and Labeling in Multilevel Database Systems

Teresa F. Lunt
Computer Science Laboratory
SRI International

For this panel on "tough issues" in database security, I was asked to comment on the issues of concurrency controls and labeling in multilevel database systems.

Concurrency Controls

There are basically two choices in implementing concurrency controls in a multilevel database system: either to implement the controls in such a way that they can be enforced without the need of trusted components (and hence free of covert channels), or to implement the controls using a trusted scheduler with global knowledge.

Covert-Channel-Free Mechanisms

The published work in this area (papers have appeared recently by Lunt, Greenberg, Downing, Tsai, Keefe, Jajodia, and Kogan) has been directed toward solutions to the former approach, in which concurrency controls can be implemented free of downward information flow channels. Although most of the work to date (with the exception of that by Lunt et al.) has considered a single central scheduler with global knowledge, that work has nevertheless sought solutions that were demonstrably free of downward information flows. Most recently, work by Bill Maimone and Ira Greenberg (to appear in the Sixth Annual Computer Security Applications Conference in Tuscon in December 1990) has shown that a central covert-channel-free scheduler with global knowledge is equivalent to a set of single-level untrusted schedulers, each with knowledge only of the information at or below its access class. This alternative, of implementing concurrency controls using only single-level untrusted subjects, is the most secure approach possible. As the published literature has shown, this approach permits any desired degree of concurrency and consistency, including strict serializability.

Trusted Mechanisms

The second alternative, that of implementing the concurrency controls using a trusted scheduler, while reducing the complexity of the functional design, increases the inherent security risk. This is because the concurrency controls are used in routine query and transaction processing, rather than merely for privileged operations performed by privileged users and authorized through a trusted path. Meaningful auditing of the covert channels introduced by such a trusted scheduler appears to be infeasible, because every query and transaction on the database will make use of the concurrency controls. As we seek increased assurance in the higher evaluation classes, the use of trusted components for routine data processing should be discouraged as introducing too much unnecessary and avoidable risk. Moreover, the interaction of the channels introduced by the database system's trusted concurrency control mechanisms and the information flows of the underlying trusted operating system may be potentially dangerous; there is currently no known theory to form a basis for reasoning about such interactions. Thus, at the higher evaluation classes, there is no way to achieve high assurance for such a design without performing a flow analysis on the combination of the trusted subjects in the database system and the operating system TCB.

Another drawback to the second alternative, in which concurrency controls are implemented by a trusted component of the database system, is that isolation of the security-relevant trusted portion of the system may become infeasible. Such isolation is crucial at the higher evaluation classes in order to demonstrate the high assurance required. Although in theory, the code that implements the database system's concurrency control algorithms could be relatively small, in practice in today's database systems, which are highly optimized for high concurrency and maximized throughput, the mechanisms involved in concurrency control are dispersed throughout the system. For example, today's database systems have numerous caches that are used for maintaining multiple versions of the data, for undo/redo logs, for recovery logs, and so on. In a distributed database system the requirements for the consistency of such structures, achieved through the application of the database concurrency controls, become even more onerous and far-reaching.

Implications for Database System Architectures

For the above reasons, for database systems that aim to meet the requirements of the higher evaluation classes, achieving covert-channel-free concurrency controls is of great importance. A design approach that requires this alternative is the so-called *constrained* or *self-contained* TCB subsetting approach. This approach segregates all code enforcing mandatory security or trusted with respect to mandatory security to a single nonbypassable TCB subset responsible for mandatory security. The database system, and hence the database system's concurrency control mechanisms, will belong to a different TCB subset, one that is constrained by the underlying mandatory TCB subset.

The difficulty of using the self-contained TCB subsetting approach is great for those database systems vendors whose processing model consists of a single server process servicing all user requests. A more natural model for a self-contained TCB subsetting approach is a processing model consisting of multiple database server instances, each servicing the requests of subjects at a single access class.

Some vendors whose processing model makes the self-contained TCB subsetting approach infeasible have argued that the self-contained approach implies reduced performance for the database system. However, at least one database system vendor includes a multi-instance processing model in a standard (not multilevel) product whose performance is comparable to that of the other vendors' products. Thus, the empirical evidence does not support this claim. More importantly, at the higher evaluation classes, tradeoffs between performance and security must be made in favor of security. Users demanding B3 or A1 database systems will not demand such high assurance frivolously; such high assurance will be demanded for extremely sensitive applications where the concern for mandatory secrecy is overriding.

In addition, the history of computer science has shown that what is ruled out as infeasible on performance grounds very often appears only a few years later as a generally accepted approach. The computer industry continues to make enormous strides in basic technology; it is quite conceivable that such progress will quickly make today's speculations and general statements about performance irrelevant.

Labels

The database-system-specific labeling concerns are threefold. First, there is the question of what granularity of labeling is desired in a multilevel database system. Second, there is the question of how to display to the user in some meaningful way the plethora of labeling information that accompanies element-level or even row-level labeling. Finally, there is an urgent need for standardization of labels among vendors' trusted operating systems. This need will extend to trusted database systems as users attempt to construct heterogeneous trusted systems from components purchased from different trusted database system vendors.

For the first question, the industry is moving in the direction of row-level labeling. This is probably adequate for the near term. However, at the same time we should experiment with systems that provide element-level labeling to determine their degree of usability (or unusability) and the extent to which they can better model the requirements of real-world multilevel applications. The decision as to the optimum granularity of labeling can only be made in such a context.

For the second question, that of how best to display voluminous label information to the user, it has been suggested that the data on the display that is at the subject's access class be highlighted, and all other data not be highlighted. This has several obvious advantages; it is immediately obvious to the user which rows or data elements the user can update (this is especially valuable in systems that polyinstantiate if a high user attempts to update low data). In addition, the user can focus on the data themselves without the distraction of a scattering of labels. This is appealing, since security should not hinder a user from focusing on the critical application at hand. Moreover, the database system need not anticipate having to display 1024 compartments for a single element or row. An obvious advantage is the ease of implementation of the approach. If the user desires to know the access class of a particular element or row, the user could "mouse" on the data, and a pop-up window could display the full access class, including the secrecy and integrity levels and all secrecy and integrity categories.

The third labeling issue, that of standardization, especially among trusted operating systems, is one that we cannot afford to overlook. This issue arises as database system vendors consider how to make their trusted database systems portable to a large number of trusted operating systems. It becomes even more pressing as users attempt to use a single vendor's trusted database system in a distributed heterogeneous environment, where each trusted operating system uses different label formats and system calls for operations (such as dominance checks) on those labels. As we eventually move to the use of heterogeneous trusted database systems in a distributed environment, the problem will become even more difficult. Thus, it is imperative that the industry begin to face the issue of standardization.

ISSUES IN TRUSTED DISTRIBUTED DATABASE MANAGEMENT SYSTEMS - A POSITION PAPER

Bhavani Thuraisingham

The MITRE Corporation, Burlington Road, Bedford, MA 01730

1. INTRODUCTION

The rapid growth of the networking and information processing industries has led to the development of distributed database management system prototypes and commercial distributed database management systems. In such a system, the database is stored in several computers which are interconnected by some communication media. The aim of a distributed database management system (DDBMS) is to process and communicate data in an efficient and cost-effective manner. It has been recognized that such distributed systems are vital for the efficient processing required in many DOD applications. For these applications, it is especially important that the distributed database systems operate in a secure manner. For example, in military applications, the DDBMS should allow users, who are cleared to different levels, access to the database consisting of data at a variety of sensitivity levels without compromising security.

A considerable amount of work has been carried out in providing multilevel user/data handling capabilities in centralized database management systems, known as trusted database management systems (TDBMS). In contrast, it is only recently that trusted distributed database management systems (TDDBMS) are receiving some attention. Note that in some DDBMSs limited forms of discretionary security controls (that is, where users access data based on authorizations) do exist.

In this paper we briefly describe the various issues involved in developing a TDDBMS. In particular, security issues in distributed architectures, query processing, transaction management, metadata management, inference problem, multilevel distributed database design, and handling heterogeneity are discussed. Note that various definitions of a DDBMS have been proposed previously. We assume the one that is most commonly used. In this definition, a DDBMS is a manager of a distributed database which is distributed across two or more nodes connected via a network. Each node can handle its local applications and participates in at least one global application. Unless otherwise stated, we also assume that the relational data model is utilized at the global and local levels.

2. ARCHITECTURAL ISSUES

The first step towards the design of any software system is to evaluate a variety of architectures and then select the most appropriate ones for the design and implementation of the system. In this section, we describe four possible architectures that have to be evaluated with respect to the various functions of a TDDBMS.

In the first of these distributed architectures, a trusted front-end machine is connected to untrusted back-end machines. Each back-end machine operates at a single level. Two data distribution schemes for such an architecture have been previously proposed. They are the following. (1) Each untrusted machine manages a database whose security level is the same as the operating level of the machine. That is, an Unclassified machine manages only the Unclassified data and the Secret machine manages only the Secret data. (2) Each untrusted machine manages databases whose security levels are dominated by the machine's operating level. That is, the Secret machine manages the Secret and Unclassified data while the Unclassified machine manages only the Unclassified data. The Unclassified data is replicated in the Secret database. This architecture does not satisfy the definition of a DDBMS as all applications are controlled by the front-end machine.

In the second of these architectures, multiple nodes are connected via a trusted network. Each node has a TDBMS which is hosted on a Trusted Computing Base (TCB). The TCB could be general purpose, or it could be specially developed for a DBMS. Furthermore, a node operates at a range of security levels and the range is not the same for all nodes. That is, one node operates at the range Unclassified to Secret, and the other node operates at the range Secret to Top Secret. The third of these distributed architectures has a configuration similar to that of the second one except that it is assumed that all of the nodes operate within the same range. In the fourth of these distributed architectures, the DDBMS is hosted on a distributed TCB (DTCB). The DTCB could be a general purpose one or it could be specially developed for a DDBMS. More work needs to be done before the issues involved in hosting a TDDBMS on a DTCB can be identified.

The TDDBMS design will no doubt depend on the architecture that is selected. From a preliminary evaluation of the various architectures, it appears that the third architecture satisfies the definition of a DDBMS and is less complex than some of the others. Therefore, the security issues that we have identified in this paper stem from the third architecture.

3. QUERY PROCESSING

The most important function of a TDDBMS is to provide a facility for users to query the distributed database system and obtain authorized and correct responses to the queries. Ideally, the distribution of the data should be transparent to the user. That is, the user should query the distributed database as if it were a centralized system. The distributed query processor, which is responsible for handling queries, should determine the locations of the various relations involved and transmit the requests to the various sites. The responses obtained from these sites have to be assembled before delivery to the user.

Following are some of the issues in secure distributed query processing.
(1) System Architecture: Two of the major components of the system architecture are the secure distributed execution monitor (SDEM) and the local TDBMS. SDEM augments the local TDBMS at each node. The distributed query processor, which is part of SDEM, is responsible for handling the queries at the global level.
(2) Semijoin Processing: Semijoin as a query processing tactic has received much attention in nontrusted DDBMSs. The various semijoin algorithms need to be adapted for multilevel applications.

(3) Polyinstantiation: Various definitions of polyinstantiation have been proposed. We assume that polyinstantiation occurs when users at different security levels have different views of the same entity. Whether polyinstantiation should be supported is still under debate. It should be noted that eliminating polyinstantiation could cause signalling channels.
(4) Multilevel Distributed Data Model: A multilevel distributed data model needs to be developed, and the views that users at different security levels could have of the distributed database at the local and global levels should be identified.
(5) Data Distribution: There are many ways of distributing the multilevel data across the various nodes. Fragmentation and replication issues have to be considered. In addition, polyinstantiation issues should also be addressed. Finally, ways of recombining the data at different security levels and nodes should be examined.

4. TRANSACTION MANAGEMENT

Transactions are managed by the Transaction Manager component of a database system. A transaction is a program unit that must execute in its entirety or not execute at all. The issues involved in transaction management are concurrency control and recovery. Concurrency control techniques ensure the consistency of the database when multiple transactions execute concurrently. Recovery techniques are necessary in order to ensure that the database is brought to a consistent state when transactions are aborted due to some failure. In order to manage transactions securely in a distributed system, it has to be ensured that the transactions associated with each local node operate securely. Although much progress has been made in incorporating transaction management features into a DBMS, providing these features in a TDBMS has only recently begun.

Following are some of the issues on transaction management in trusted centralized and distributed database systems.
(1) Security policy: The security policy enforced by the system needs to be extended for transaction management. An important issue that needs to be addressed is whether a transaction operates at a single level during its execution or whether it can change security levels during execution. If it is the former, then the transactions are single-level. If it is the latter, then the transactions are multilevel.
(2) Serializability condition: Appropriate serializability conditions for local and global transaction management have to be formulated. These conditions should ensure that only serializable schedules are executed.
(3 Concurrency control: The various concurrency control techniques need to be examined for transactions which operate in a multilevel environment. These techniques should ensure that not only is the consistency maintained, but the actions of higher level transactions do not interfere with lower level ones. In addition, deadlocks and starvations have to be prevented. The three techniques that are being investigated are locking, timestamping and validation. The locking technique has a potential for a covert channel. Some suggestions have been given for adapting the locking technique using multiple versions of data. No satisfactory technique has yet been proposed which would satisfy all the requirements. Therefore, this is an area where much research remains to be done.
(4) Recovery: Recovery techniques should ensure that the system should be recovered to a consistent state in the event of any failure. Techniques for extending the two-phase commit protocol for recovery in a TDDBMS have been proposed. However, issues on handling network partitions have yet to be investigated. Extensions to the three-phase commit protocol for handling partitions in a TDDBMS need to be proposed.
(5) Constraints: Techniques for handling integrity and security constraints during transaction management in a multilevel environment have not received any attention. There could be a potential for signalling channels, covert channels, and security violations via inference when the constraints have to be processed during transaction execution. More research needs to be done in this area.
(6) Performance: For many military applications, it is important that not only are the transactions executed securely, but also that they have high performance. Therefore, simulation studies of transaction management in a multilevel environment should be carried out. In the past, simulation studies have been shown to be extremely cost-effective methods for modelling the behavior of distributed database systems. By carrying out simulation studies, inefficient implementation efforts can be avoided.

5. METADATA MANAGEMENT

To function effectively, information about the various databases, users, network, and the administrative features of the distributed system must be integrated. A metadatabase (sometimes referred to as the data dictionary) is the repository of such information. The module which manages the metadatabase is called the Metadata Manager (or Data Dictionary Manager). The function of the Metadata Manager is to ensure the consistency of the metadatabase and to provide the support in terms of metadata access to the other functions of the DDBMS.

Many of the issues involved in metadata representation and management have been investigated for centralized systems. Further, the Information Resource Dictionary System (IRDS) has been proposed as a standard for representing and managing the metadata. However, little work has been accomplished on managing the metadata in a distributed environment. The security impact on metadata management has also not received much attention even in centralized systems. Only a few of the designs have addressed the issues involved to some extent. In order to investigate the security impact on metadata management in a distributed environment, the following topics should be addressed first:
(1) Metadata Management in a DDBMS, and (2) Metadata Management in a TDBMS.

The TDDBMS functions requiring metadata access can be grouped into three categories. They are (1) functions for processing distributed database user requests, (2) functions for processing system designer requests, and (3) functions for processing system administrator requests. As in the case of a DDBMS, the functions for processing distributed database user requests in a TDDBMS are transaction analysis, distribution, access control and translation. The metadata required by these functions includes mappings, schemas, integrity constraints, discretionary security constraints, and transaction logs. The functions for processing system designer requests in a TDDBMS include performance evaluation, file conversion, and file allocation. These functions require information on file access programs, total volume of queries for each file, total volume of updates for each file, the security levels of the files, the security levels of the user who requests queries and updates on files, and the number of polyinstantiated tuples in a fragment of a relation and database schema. The administrators of a TDDBMS are the network administrators, global and local database administrators, and global and local system security officers. The functions of the network and database administrators are the same for a DDBMS and a TDDBMS. The functions of the system security officer are to design, enforce, and monitor the security features of the system. A TDDBMS must also support mandatory security constraints (which assign security levels to the data) in addition to the integrity constraints and discretionary security constraints. The mandatory security constraints may be used by the Transaction Analysis and Access Control functions for operations such as query modification. The system security officers are responsible for creating and maintaining the security constraints at the local and global levels. The consistency and the completeness of the constraints should also be ensured.

It should be noted that the metadata itself could be assigned different security levels. For example, the metadata could be maintained either at system-low, system-high, or at different security levels. Some implications of each of these configurations for a TDDBMS are as follows. When metadata is stored at system-low, there is a trusted manager at each security level. The manager at level L in node 1 communicates with the manager at level L in node 2 to retrieve the metadata stored in a remote machine. A disadvantage of storing the metadata at system-low is that the information required by higher level users is also stored at system-low. When metadata is stored at system-high, there is one trusted manager operating at system high. The managers at different nodes communicate in order to access remote metadata. When metadata is multilevel, there is a manager at each security level. The managers do not have to be trusted. The manager at a security level L in node 1 communicates with the manager at security level L in node 2 in order to access remote metadata. Some integrity problems with multilevel metadata have been identified previously.

Various schemes have also been proposed to store the metadata. They include having the metadata (1) centralized at one node, (2) fully replicated at each node, and (3) distributed across various nodes with possible partial replication. If the metadata is centralized, then it can be kept consistent. Also the users will know exactly where to look for the metadata. However, if the node which stores the metadata fails, then the metadata cannot be accessed. Furthermore, if the system is congested, then access to the metadata will be difficult. Fully replicating the metadata overcomes the problems of failures and congestion. However, the various copies of the metadata have to be kept consistent. Distributing the metadata with possible partial replications eases the consistency problem. However, the locations of the metadata have to be determined before they can be accessed. Usually, the metadata which describes the data at a node is also stored at the same node. If a data fragment is replicated in several nodes, then the metadata associated with that fragment could be stored at the site where the primary copy of the fragment is stored. If all of the metadata is assigned the same security level, then the techniques used to allocate metadata in a DDBMS can be used for a TDDBMS also. If the metadata is assigned multiple security levels, then the data distribution schemes used for a multilevel distributed database can be applied to allocate the metadata also.

6. INFERENCE PROBLEM

The word "inference" is commonly used to mean "forming a conclusion from premises," where the conclusion is usually formed without expressed or prior approval, that is, without the knowledge or consent of anyone or any organization that controls or processes the premises or information from which the conclusion is formed. The resulting information that is formed can be innocuously or legitimately used or it can be used for clandestine purposes with sinister overtones. The term "information" is broadly defined to include raw data as well as data and collections of data which are transformed into knowledge. The inference process becomes a problem when unauthorized conclusions are drawn from authorized premises.

Two distinct approaches have been proposed for handling the inference problem. They are: (1) Handling of inferences during database design. Here, the security constraints during database design are handled in such a way that security violations via inference cannot occur. (2) Handling of inferences during query processing. Here, the query processor is augmented with a logic-based *Inference Engine*. The *Inference Engine* will attempt to prevent users from inferring unauthorized information. We believe that inferences can be most effectively handled and thus prevented during query processing. This is because most users usually build their reservoir of knowledge from responses that they receive by querying the database. It is from this reservoir of knowledge that they infer unauthorized information. Moreover, no matter how securely the database has been designed, users could eventually violate security by inference because they are continuously updating their reservoir of knowledge as the world evolves. It is not feasible to have to redesign the database simultaneously.

Handling inferences in a distributed environment is more complex. One possible approach is to augment each node with an inference controller component. This component is hosted on top of the distributed execution monitor. The various inference controllers have to communicate in order to handle inferences. We could also envisage having a few dedicated inference controllers connected to the network whose functions are only to determine the inferences that users could draw. More research needs to be done in order to determine the feasibility of handling inferences in a distributed environment.

7. MULTILEVEL DISTRIBUTED DATABASE DESIGN

Designing a multilevel distributed database includes the following two steps:
(1) Designing the multilevel database - This design is carried out as in the case of a TDBMS. That is, the security and integrity constraints are examined, and the schema is assigned appropriate security levels. Tools need to be developed in order to design the schema for a multilevel database. Suggestions for multilevel database design have been given previously. However, no attempt has yet been made to develop viable tools for multilevel database design. Future work should focus in this direction.
(2) Designing the global, fragmentation, allocation, mapping, and physical schemas - Once the multilevel database is designed, distribution and allocation issues are addressed as in the case of a DDBMS. That is, the following schemas should be designed:

- Global schema: The entire distributed database is regarded as a single database, and it is designed at the conceptual and logical levels. Different data models could be used to represent the data at the conceptual and logical levels.
- Fragmentation schema - Each database relation could be fragmented into several components. The fragmentation schema includes a description of the various components of a relation.
- Allocation schema - The distributed database is scattered across several nodes. The allocation schema describes where the relations and fragments of relations are stored.
- Local mapping schema - This consists of mappings between the global and local representational models.
- Physical schema - The internal schema of the local databases.

Many of the issues involved in designing the global, fragmentation, allocation, mapping, and physical schemas are the same for both multilevel and nonmultilevel applications. There are various steps involved in designing a multilevel database. The first step in the design process is to capture all of the information present in the application. In order to do this, an appropriate data model has to be used. If there is no single data model that will capture the information present in the entire application, then a combination of data models have to be used. The information for a multilevel application will include the integrity constraints, security constraints, structural relationships, and semantic relationships between the various entities. The second step is to ensure that all of the information present in the application has been captured by the information model. Various techniques that have been proposed to determine the completeness of the knowledge base can be used for this purpose. The first and second steps are repeated until the designer is convinced that all of the information is captured. The third step is to translate the information captured by the data model into an appropriate format if necessary in order to process it efficiently. The fourth step is to analyze the data and subsequently generate the database schema. The fifth step is the testing phase where the output generated is validated. For applications which are involved with very large amounts of data it is useful to implement a rapid prototype of the system in order to carry out the validation. The fourth and fifth steps are repeated until satisfactory output is obtained.

8. HANDLING HETEROGENEITY

The ultimate solution to handling heterogeneity is to adopt a standard which each local system in the distributed environment should follow. However, vendors, eager to maintain their advantages over competitors to preserve their share of the market, discourage the adoption of a universal standard. This is because compromise is inevitable when adopting a standard which can lead to diminution of individual performance and erosion of customized advantages and efficiency. At least for the foreseeable future, different database systems with their differing data representation schemes are here to stay. Consequently, in order to reconcile the contrasting requirements, tools which enable users of one system to use other systems are necessary. Therefore efficient solutions for interconnecting different database systems as well as administering them, have to be provided. However, the increasing popularity of heterogeneous DDBMSs should not obscure the need to maintain security of operation. That is, it is important that such a system operate securely in order to overcome any malicious corruption of data as well as prohibit unauthorized access to and use of classified data, especially with military applications. Incorporating security into the operation of a heterogeneous distributed database system brings about a complexity not present in homogeneous systems. This is because it is not straightforward to transform the security properties of one system into those of another.

Various types of heterogeneity can be identified. Two of the more important ones are secure data model heterogeneity and heterogeneity with respect to local TDBMS designs. To handle secure data model heterogeneity, mappings have to be developed which transform the constructs of one secure data model into those of another. Therefore, if there are n different data models, n**2 (n squared) different translators are necessary. This is not desirable. One useful approach is to define a secure unified data model for the global view. Then, mappings are necessary only from (to) the unified data model to (from) a specific data model. Therefore, in this approach, if there are n data models, only 2n translators are necessary. When there is heterogeneity with respect to local TDBMS designs, not all the local TDBMSs are designed using the same methodology. Therefore, the various designs have to be made compatible at the global level. It should be noted that TDBMSs based on various designs are being developed commercially. For certain applications, it may be necessary to interconnect these systems.

9. OTHER ISSUES

In this section, we discuss some of other issues that have to be addressed. They are network security issues, steps to developing a TDDBMS, and developing new generation TDDBMS.

The major issues in DDBMS security are (1) the secure operation of the various nodes, and (2) the secure operation of the network which interconnects the nodes. Much of the previous work on TDDBMS has focused only on the first aspect. It assumes that the network is multilevel secure. Future work should include the investigation of network security issues. In particular, various network architectures need to be investigated for a TDDBMS. These architectures could include those based on a token-ring network, a token-bus network, and point-to-point interconnection.

The steps required to develop a TDDBMS depend on the level of assurance that is expected of the system. In addition to the design and implementation steps (which will consist of the design and implementation of the models for query processing, update processing, transaction management, and metadata management), a TDDBMS development process would include the following: (1) Security Policy: This is a discussion on the policies for mandatory security, discretionary security, integrity, auditing, authentication, accounting, and the roles of the distributed database administrator, the local database administrators, and the systems security officer. (2) Model: A security model for the distributed database system should be developed. There are three components to this model. They are: model of the local TDBMS, model of the network, and model of the secure distributed execution monitor. (3) Specification: Security requirements expressed by the model are specified (either formally or informally), and the correspondences between the specification and the model and the specification and the implementation are demonstrated.

Recent work on DDBMSs has included (1) exploiting parallel architectures, (2) extending relational systems to support deductions and complex objects, and (3) developing non-relational systems such as object-oriented distributed database systems. In order to conduct similar research for TDDBMSs, it has to be first carried out for a TDBMS. That is, the issues on (1) using parallel database architectures for designing a TDBMS, (2) extending the multilevel relational model to support deductions and complex objects, and (3) developing an object-oriented TDBMS need to be investigated first. The work can then be extended to a secure distributed environment.

10. CONCLUSION

In this paper we have described the need for TDDBMSs and identified some security issues that need to be given further consideration. We have also provided solutions to some of these. More research, prototype development, and simulation studies need to be carried out before useful TDDBMSs can be developed successfully.

ACKNOWLEDGEMENT AND DISCLAIMER

We gratefully acknowledge the Department of the Navy (SPAWAR) for supporting our work on identifying the issues in a TDDBMS under contract F19628-89-C-0001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Navy or the United States Government.

SYBASE: The Trusted Subject DBMS

Helena B. Winkler-Parenty

Sybase, Inc.
6475 Christie Avenue
Emeryville, CA 94608

Historical Context

In 1983 the Department of Defense (DoD) published the Trusted Computer System Evaluation Criteria (TCSEC) [1]. It is a description of the varying levels of trust that can be placed in a computer system, based on whether the system meets a comprehensive set of standards for features and assurances. The TCSEC was republished as a DoD standard, Dod-5200.28-STD, in December, 1985. The Trusted Network Interpretation (TNI) [2], an interpretation of the TCSEC for networks, was published in 1987. The Trusted Database Interpretation (TDI) [3], an interpretation of the TCSEC for database management systems (DBMS), is expected to be published by the end of 1990.

The TDI

In addition to providing information specific to databases, the TDI contains a section describing architectural approaches and evaluation strategies for building trusted systems. The purpose of this section is to provide guidance on evaluating systems when the system is comprised of several pieces that were developed independently, perhaps by different vendors. The techniques presented can be used beneficially to structure an operating system and its utilities, a DBMS running on an operating system, or any software system.

The TDI presents a strategy meant to facilitate evaluation of computer systems, which is to divide the Trusted Computing Base (TCB) into a collection of parts, called TCB subsets. Each subset contains its own set of subjects, objects, and an access control policy that it enforces. Subsets must mediate every access of a subject to an object. In addition, they must be self-protecting. Every subset is evaluated individually. The evaluation requirements for the whole TCB depend upon the placement of system components into subsets, and the privileges associated with each subset.

Trusted Subject Overview

There are two architectural approaches presented in the TDI that are particularly significant. These approaches are TCB augmentation using trusted subjects and hierarchical TCB subsets. TCB augmentation using trusted subjects allows a TCB subset's MAC policy to be extended by the addition of a trusted subject to the total TCB.

A trusted subject is a process that contains its own MAC policy, that it enforces on its own subjects and objects. This means that it is trusted to operate over a range of security levels. An example of this is a trusted DBMS that performs MAC and DAC on DBMS objects, running on an operating system that performs MAC and DAC on operating system objects. See figure 1.

A trusted subject can be placed into a TCB subset that already exists, i.e. the operating system's TCB subset, or it can be put into its own TCB subset. If the trusted subject is placed into a pre-existing subset, a complete evaluation is performed on this TCB subset. The trusted subject extends the MAC policy of this pre-existing TCB subset.

If the trusted subject is placed into its own subset, then only the new subset is fully evaluated. For B2 and above systems global analysis, such as penetration testing and covert channel analysis, will be performed on both the new TCB subset containing the trusted subject and the pre-existing TCB subset whose MAC policy is extended.

With the hierarchical TCB subsets approach the MAC policy for the entire TCB is in one subset. For example, the operating system performs MAC and DAC on operating system objects, and the DBMS only performs DAC on DBMS objects. See figure 2.

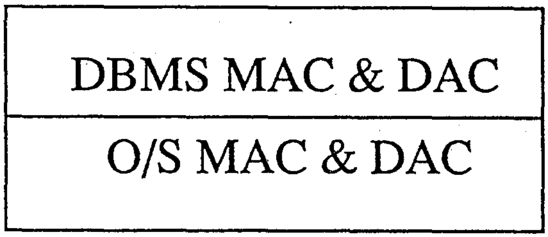


Figure 1: Trusted Subject

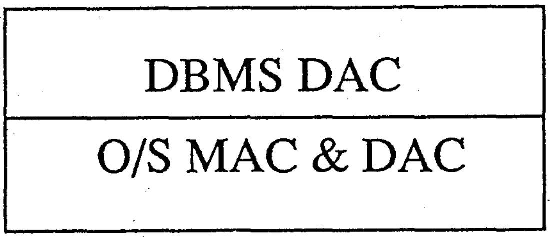


Figure 2: Hierarchical TCB Subsets

Sybase's Approach

An example of a trusted DBMS that uses this approach is the SYBASE Secure SQL Server^{TM}. The SYBASE Secure SQL Server is a relational DBMS targeted at the B1 level of trust. The SYBASE Secure SQL Toolset^{TM} is a set of user interface tools and utilities. The Secure SQL Server, the Secure SQL Toolset, and the operating system(s) on which they run, are all part of the total TCB.

System Architecture

The Secure SQL Server and the Secure SQL Toolset make use of the client/server architecture. In this model, clients make requests and process the server's responses. Servers respond to these requests and return data and other information to the clients. The Secure SQL Server and Secure SQL Toolset can run on the same, or different, machines. The operating system runs in one TCB subset and the Secure SQL Server runs in another TCB subset. The Secure SQL Toolset runs as an ordinary user process.

The Secure SQL Server uses a multi-threaded architecture. The Secure SQL Server runs as a single operating system process. For each DBMS user the Secure SQL Server creates an internal task. The Secure SQL Server handles scheduling, task switching, disk caching, locking, and transaction processing for these DBMS tasks. Users can communicate with the Secure SQL Server either from the Secure SQL Toolset or from an application program.

Identification and Authentication

Identification and authentication takes place as follows. (To simplify this discussion, only logging in to the Secure SQL Toolset will be discussed. Logging in from an application program would proceed in a similar fashion.) The user is prompted for his or her name and password. The default value for the user's DBMS user account name is his or her operating system account name.

If the user is logging in from a multi-level secure (MLS) operating system, the Secure SQL Toolset obtains the user's operating system login level from the MLS operating system. Otherwise, the user is prompted for the DBMS login security level. The user's name, password, and login security level is then inserted into a login packet and passed from the Secure SQL Toolset to the Secure SQL Server for authentication. The underlying operating system(s) and the network are relied upon to safely transmit the packet. No MLS functionality is required, only integrity.

When the Secure SQL Server receives the packet, the user's name is checked to confirm that there is a valid user account with the same name. In addition, it checks that the account has not been locked, a mechanism by which user accounts stay valid but cannot be used. This mechanism is useful when steps need to be taken so that a user cannot log into his or her account for security or administration-related reasons, such as when a user goes away on vacation.

The password is encrypted with a one-way encryption algorithm and compared with the encrypted password stored in *syslogins*, a system catalog. The user's login security level is compared with his or her maximum allowable login security level, also stored in *syslogins*. The user's maximum allowable login security level must dominate the requested login security level.

If all of these checks succeed, the login attempt was successful and a DBMS task is created on behalf of the user. Otherwise, the user gets back the error message: Login failed. The DBMS task created on behalf of the user is quite similar in both concept and content to the user process that is created when a user logs into an operating system.

Operating System Interactions

The description of identification and authentication typifies the interactions between the Secure SQL Server and the operating system it runs on. The Secure SQL Server enforces its own security policy. It does not rely upon the operating system for security functionality or assurance, and uses only low-level operating system services such as I/O.

The Secure SQL Server enforces mandatory and discretionary access controls on DBMS objects, just as operating systems enforce mandatory and discretionary access controls on operating system objects. The Secure SQL Server enforces mandatory access control (MAC) on rows and discretionary access control (DAC) on tables and databases. The Secure SQL Server extends the MAC policy of the underlying operating system. Because the objects upon which the operating system and the Secure SQL Server operate on are different, there is no conflict between the two policies. The operating system does not mediate access to DBMS objects and the DBMS does not mediate access to operating system objects.

In addition to providing MAC, DAC, and identification and authentication, the Secure SQL Server has been designed to meet all other TCSEC requirements at the B1 level of trust.

Site Security

Although the operating system and DBMS security policies are disjoint, care must be taken to ensure that all site security requirements are met. Specifically, because the DBMS stores its data in operating system objects, the operating system must be configured to protect this data. There are three separate mechanisms that can be used. DAC can be used to ensure that only the DBMS process has access to the operating system objects that contain the database. MAC can also be used either by labeling the operating system objects at the highest level at which data might be stored within the database, or by allocating a special compartment for these objects and the Secure SQL Server process. Finally, the machine on which the Secure SQL Server runs can be configured as a database machine. This means that no users other than trusted system administrators are allowed access to this machine.

Secure SQL Toolset

Unlike the Secure SQL Server, which is responsible for virtually all of the DBMS security policy, the Secure SQL Toolset is only responsible for correctly implementing its portion of identification and authentication. Because the Secure SQL Toolset does not enforce a mandatory access control policy, it is not a trusted subject. Therefore, even though the Secure SQL Toolset is trusted with respect to the Secure SQL Server and is part of the total TCB, it is not trusted with respect to the operating system on which it runs.

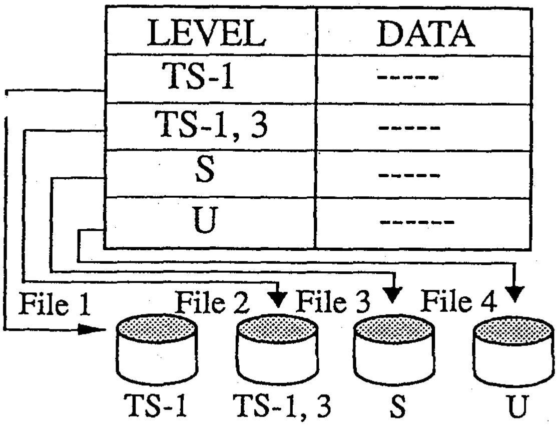
Comparison to Hierarchical TCB Subsets

Trusted subjects is a natural approach to building trusted systems. It allows each component of the system to enforce a portion of the overall system's security policy on the objects that it knows best. By allowing each portion of the system to enforce its own security policy, the end result is a system that more closely meets the needs of its users. For example, a key component of a DBMS is users' ability to specify how the data will be stored in the database, based upon its semantic content. Clustered indexes provide this ability, and their implementation is the same in the Secure SQL Server as in the standard SQL Server product. See figure 3.

In contrast, when a DBMS is built based on hierarchical TCB subsets, the DBMS is constrained by the mandatory access control policy of the underlying operating system. Since operating systems enforce MAC on files, the DBMS must store its data in files having security levels that match the level of the DBMS data. For example, a row labeled Top Secret-NATO would have to be stored in a file labeled Top Secret-NATO. In this system, although the user may want to have data that semantically belongs together stored together, this can only occur if the data happens to have identical security levels. See figure 4. Not only does the data have to be stored in several files, but any index on this table also has to be stored in several files, one for each level of data. This means that for data which needs to be returned in the order of the index, an additional sort will be required to merge the data into the proper order.

LEVEL	DATA
TS-1	-----
TS-1, 3	-----
S	-----
U	-----

Figure 3: Trusted Subject



LEVEL	DATA
TS-1	-----
TS-1, 3	-----
S	-----
U	------

Figure 4: Hierarchical TCB Subset

Another possible implementation is to allow only single-level tables, but multi-level views. Views can then be used to create the illusion of multi-level tables. Additional system overhead is required to combine many tables to form a multi-level view. Again, data that semantically belongs together cannot be stored together to optimize performance.

In addition to the performance degradations already mentioned, all system resources which could contain data at multiple levels will need to be divided, one instance of each system resource for each level of data. For example, the buffer cache will need to be subdivided, and information on tasks and locks will need to be managed separately. All of these will result in less system resources being available for any specific security level.

Conclusion

Trusted subjects is a powerful approach to building high performance trusted systems. It allows a system architect to build on top of a pre-existing TCB, extending the TCB's mandatory access control policy so that it can accommodate the requirements of a trusted application or utility. This means that a DBMS built with trusted subjects will perform better than a system built with hierarchical TCB subsets.

References

[1] "Trusted Computer System Evaluation Criteria," Dept. of Defense, National Computer Security Center, Dec. 1985.

[2] "Trusted Network Interpretation," Dept. of Defense, National Computer Security Center, July 1987.

[3] "Trusted Database Management System Interpretation," Dept. of Defense, National Computer Security Center, March 1990.

NCSC "Tough Issues" Panel
Constrained Trusted Computing Base Subsets

Linda L. Vetter
Director, Secure Systems
Oracle Corporation

Introduction

Oracle Corporation is developing a multilevel secure (MLS) relational database management system (RDBMS) product for commercial and government markets. Oracle's MLS RDBMS architecture is unique among MLS database systems available from or under development by other vendors because Oracle's MLS RDBMS, Trusted ORACLE RDBMS, utilizes a "constrained" trusted computing base (TCB) subset architecture within its fundamental design.

Producing an MLS RDBMS using a constrained TCB subset approach is not a simple task. However, the constrained TCB subset architecture promises implementation benefits to the secure RDBMS user, as well as benefits to the database vendor, evaluators, and accreditors, that make it a logical choice of system architecture where the underlying DBMS architecture supports this option.

Constrained TCB Subset Approach

In a constrained TCB subset architecture, there is a clear definition between the RDBMS portion of the trusted computing base (TCB) and the operating system (OS) and hardware portions of the TCB on which the RDBMS runs. The constrained TCB subset approach extends security features already provided by a trusted operating system with complementary security capabilities in the RDBMS; the RDBMS TCB implements the DBMS-specific aspects of the security policy without superseding the security policy of the secure operating system base.

This architecture efficiently utilizes the security mechanisms of each component of the trusted computing base in a non-redundant manner to form a unified, secure computing system. The TCB is hierarchically layered into defined subsets, each subset enforcing a specific part of the overall system security policy. A subset at a higher layer of the TCB expands upon the security policies enforced by lower layers but cannot violate or supersede the security policies of the lower layers. A layered, constrained TCB subset architecture is consistent with fundamental principles of sound software and system engineering, as well as with the high assurance security concepts of structured, layered components.

A DBMS must meet specific requirements to qualify as a constrained TCB subset. A constrained subset must function untrusted with respect to any underlying TCB subsets of the system. In other words, a constrained TCB subset cannot include trusted subjects that are allowed to operate across a range of sensitivity levels, nor can it run with any privileges that would allow it to bypass or violate the access control policies of any more primitive subset of the TCB. By constraining its subjects to operating wholly within the bounds of the OS security policies, the DBMS and underlying operating system can each be completely evaluated independently.

The Case for Constrained TCB Subsets

A constrained TCB subset architecture is a logical approach to MLS RDBMS secure system design. The constrained TCB subset approach provides for reuse and extension of existing evaluated trusted systems. This allows secure RDBMS vendors, especially those dedicated to providing portable solutions, to leverage existing systems on the evaluated products list (EPL) and thus populate the EPL with secure RDBMS products on multiple platforms more quickly. The availability of numerous evaluated RDBMS products on the EPL, in turn, gives users greater flexibility in choosing a secure system configuration while providing products with a high degree of security assurance.

A portable MLS RDBMS with a constrained TCB subset architecture should also take less time to maintain, enhance and port because a new mandatory TCB does not have to be designed and implemented with regard to each hardware platform. Also, the constrained TCB subset approach allows vendors to build independent products to extend a system's TCB to enforce additional DBMS-specific security without having to re-verify OS-level security.

Benefits of Constrained TCB Subsets

Ease of evaluation: The TCB subset architecture exploits the security features of an existing, evaluated OS. Since the operating system and hardware base have already undergone evaluation, only the RDBMS TCB subset (and its interfaces to the operating system) must be evaluated. This makes for simpler, shorter evaluations. This entire concept allows trusted DBMS products to be available from vendors other than the OS vendors, and it leverages off of all the past and future work of vendors and the NCSC in populating the evaluated products list (EPL).

The benefits of reduced evaluation efforts extend to multiple OS platforms as well. For each new platform to which the RDBMS is ported, only the implementation of the interface from the RDBMS to the operating system changes. Subsequent evaluations of the same version of the RDBMS at the same security class, on different platforms at the same security class, are therefore even further reduced in scope.

Portability: The layered structure of the constrained TCB subset architecture ensures clean interfaces and facilitates portability of trusted products. The ORACLE RDBMS is already known for its portability to a variety of hardware and software bases, so a security architecture that supports portability is a natural fit for ORACLE. In addition, ORACLE's multi-server RDBMS design gave Oracle Corporation great flexibility in selecting a security implementation, where single-server architectures tend to be limited to dedicated machine and trusted subject implementations. As many major operating system vendors are actively producing secure operating systems for evaluation, a portable MLS RDBMS architecture offers customers a choice of many potential hardware and software platforms, including micros, minis, workstations and mainframes. Since the secure RDBMS builds upon security features of the platform, it is also adaptable to additional environments (e.g., compartmented mode workstations or "international" Orange Book variations).

Appropriateness of policy enforcement: A constrained TCB subset architecture presents a "seamless" view of computer security to users by providing each aspect of security functionality and granularity at appropriate levels, for example: logon security at the system access level, file security at the operating system level, and database table and view security at the RDBMS level. This approach also produces a system with comparable functionality and usability of standard, full function products combining, for example, a highly functional RDBMS and a general purpose operating system.

Constrained TCB Subsets Considerations

The primary difficulty in implementing a constrained TCB subset architecture is eliminating DBMS privileges that bypass operating security mechanisms. For example, in most DBMSs a read operation requires locking the data. Locking the data of a lower level to allow read down requires write access to a lock at the lower level; this introduces a covert channel which violates mandatory security policies. Eliminating this "secure readers-writers" problem requires a concurrency control mechanism that can provide pure read capabilities and avoids the use of locks.

A drawback of constrained TCB subsetting is that multilevel database objects must physically be stored in multiple operating system segments or files. This is required in order to utilize the operating system mandatory security policy and the enforcement mechanisms for database MAC enforcement. While it does mean that for all inserts and updates (which are by nature single level) and for all single level queries this approach is more efficient (no row-by-row label storage or checking), reading a multilevel object requires retrieving portions of the data from each different level.

Why Oracle Chose TCB Subsets

There are two primary reasons why Oracle Corporation chose to incorporate a constrained TCB subset architecture into its Trusted ORACLE RDBMS product: 1) market viability; and 2) technical feasibility.

Product acceptance, and therefore market viability, necessitate that a product be designed to meet the needs of users. Oracle Corporation has been successful to date in the database management system market by providing well-designed RDBMS products that are integrated, portable, and highly functional. Oracle believes that using a constrained TCB subset architecture, as described below, will best allow it to produce MLS

RDBMS products that build upon the traditional strengths of its product line.

Oracle Corporation's ability to implement a constrained TCB subset architecture in its MLS RDBMS is unique among database vendors due to the multi-server architecture and multi-versioning capabilities of ORACLE RDBMS, upon which Trusted ORACLE is based.

For example, when ORACLE updates a row, it also records enough information to generate a pre-update replicate of the row. Replicates are stored in memory or in the rollback segment areas of the database. Rollback segments are used for read consistency to ensure that a query uses a consistent image of the database at the start of the query. In a multilevel secure environment, multi-versioning technology allows users at various security levels to read data at lower security levels without locking data, thereby eliminating covert channels. Thus, ORACLE's multi-versioning concurrency control mechanisms eliminate the secure readers-writers problem described in the preceding section.

The inherent characteristics of a constrained TCB subset architecture will allow Trusted ORACLE RDBMS users to take maximum advantage of the latest in RDBMS technology, such as support for symmetric multiprocessing machines, on-line transaction processing (OLTP) environments, new read-only media, and distributed applications in heterogeneous environments.

ORACLE already provides features that readily lend themselves to the constrained TCB subset architecture. The ORACLE RDBMS is designed for portability; this includes years of defining and refining the distinct operating system interfaces so necessary in TCB subset architecture. In addition, the ORACLE RDBMS TCB subset approach provides full functionality, including fully ANSI-compatible SQL and referential integrity, as well as versatile features such as roles and views. In Trusted ORACLE, many existing RDBMS capabilities are being utilized, such as multiple servers, multi-versioning and distributed database support. Utilizing the existing, flexible RDBMS architecture and base features also ensures high compatibility and application portability across all versions of ORACLE and Trusted ORACLE.

Oracle preferred to provide a constrained TCB subset implementation rather than limiting the product to only a single trusted subject design for several reasons:

First, typical trusted subject designs require that the DBMS re-implement and bypass certain mandatory access control (MAC) facilities typically enforced by the operating system. The DBMS processes that supplant the operating system MAC mechanisms must execute as trusted subjects with special privileges. A single-server DBMS trusted subject typically must perform operations in violation of the operating system security policy, for example apparent downgrading of data to subjects at lower security levels.

Use of a trusted subject architecture can limit OS functionality and thereby limit system functionality. Some features usually provided by the operating system must be incorporated into the DBMS, often implementing the same or similar security model, leading to additional code to evaluate and maintain.

Summary

Each approach to database security involves trade-offs. After weighing the pros and cons of each approach, Oracle Corporation has chosen to use the constrained TCB subset architecture in its MLS Trusted ORACLE RDBMS product. The availability of this approach to Oracle, due to its unique DBMS architecture, and the benefits of this approach relative to other approaches (including reduced evaluation time, exceptional portability, high functionality, compatibility with existing products, interoperability, and configuration flexibility) lead Oracle to believe that the benefits of constrained TCB subsets substantially outweigh any disadvantages.

MULTILEVEL OBJECT-ORIENTED DATABASE SYSTEMS

(Panel Session)

Ravi Sandhu

Department of Information Systems and Systems Engineering
George Mason University, Fairfax, VA 22030-4444

There is a consensus that object-oriented database management systems will dominate the next generation of commercial products, much as relational database management systems are dominant today. Several approaches to multilevel security for object-oriented systems have been proposed. The underlying assumptions adopted by each one and their motivating forces are somewhat different. This makes a relative comparison difficult since different assumptions and motivations inevitably lead to different design trade-offs.

This panel brings together some of the leading researchers in this arena. Each panelist will present a brief overview of his or her approach, perspective and insights. The panelists have been requested to address the following questions at some point during their presentation.*

1. How much trusted code is required for their architecture? How much reuse of commercial off the shelf software is feasible?
2. How do their systems enforce inheritance? Can the inheritance mechanisms of their systems be used to define and enforce security policies?
3. Do their systems support security policies for composite objects and multilevel composite objects as defined in Catherine Meadows' position paper?

Each panelist has contributed a position paper which is included in these proceedings.

A biographical note on each of the panelists and the panel session chairman is given below (in alphabetical order).

- **Sushil Jajodia** is currently Professor of Information Systems and Systems Engineering at the George Mason University, Fairfax, VA. He is also Principal Scientist in the Security Technical Center at the MITRE Corporation in McLean, VA. Prior to joining GMU in 1988, he directed the Database and Expert Systems Program at the National Science Foundation. He earlier headed the Database and Distributed Systems Section at the Naval Research Laboratory, Washington and was an Associate Professor of Computer Science and Director of Graduate Studies at the University of Missouri, Columbia. He received the Ph.D. from the University of Oregon, Eugene. His current research interests include information systems security, database management and distributed systems, and parallel computing. He has published more than 60 technical papers in refereed journals and conferences and has co-edited three books. He is on the editorial board of the IEEE Transactions on Knowledge and Data Engineering. He

*I am indebted to Catherine Meadows for questions 2 and 3, which are posed in her position paper.

- **Ravi Sandhu** is currently an Associate Professor of Information Systems and Systems Engineering at the George Mason University, Fairfax, VA. Prior to that he was an Assistant Professor of Computer Science at the Ohio State University, Columbus. He earlier held R&D and teaching positions at the Indian Institute of Technology, Jawaharlal Nehru University and Hindustan Computers Limited all in New Delhi, India. His research interests include information systems security, multilevel database management systems, models and mechanisms for integrity, and secure distributed systems. He received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Bombay in 1974, the M.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Delhi in 1976, and the M.S. and Ph.D. degrees in Computer Science from Rutgers University in 1980 and 1983 respectively. Ravi Sandhu has authored numerous journal and conference publications on information security. He is the program chairman for the 1991 IEEE Workshop on Computer Security Foundations.

- **Bhavani Thuraisingham** is a lead engineer at the MITRE Corporation. Her current research interests are in database security and the applications of mathematical logic in computer science. Her recent research contributions include security in distributed database management systems, secure object-oriented data models, logic for secure data/knowledge base management systems, techniques for handling the inference problem, and the complexity of the inference problem. She is also leading two team efforts on the design and implementation of a trusted distributed query processor and a database inference controller. Previously Dr. Thuraisingham was at Honeywell Inc. where she was involved with the design of Lock Data Views, and before that at Control Data Corporation where she was involved with the development of CDCNET. She was also an adjunct professor and member of the graduate faculty in the Department of Computer Science at the University of Minnesota. Dr. Thuraisingham received the M.S. degree in computer science from the University of Minnesota, M.Sc. degree in mathematical logic from the University of Bristol, England, and the Ph.D. degree in recursive functions and computability theory from the University of Wales, Swansea, United Kingdom. She has published over 50 technical papers including over 25 journal articles in database security, distributed processing, AI and computability theory. She is a member of the IEEE Computer Society and ACM.

has edited special issues of the IEEE Transactions on Software Engineering, Journal of Systems and Software, and Bulletin on Data Engineering. He is general co-chair of the 2nd International Symposium on Databases in Parallel and Distributed Systems and program chair of the 4th IFIP Working Group 11.3 Workshop on Database Security. He is a member of the IEEE Computer Society Publication Planning Committee, and has chaired the IEEE Technical Committee on Data Engineering for two years. He is a senior member of the IEEE Computer Society and a member of ACM.

- **Teresa F. Lunt**, of SRI's Computer Science Laboratory, is in charge of computer security research at SRI, where she is leading two landmark programs: the SeaView multilevel secure relational database system and the IDES Intrusion-Detection system. She is also leading a new research area in security for knowledge-based systems and using AI techniques for computer security. Prior to joining SRI in early 1986, she worked at the MITRE Corporation for four years and later at SYTEK's Data Security Division for two years. She has worked on audit trail analysis, automated security guards, security models, and formal verification of secure systems. She received the A.B. degree from Princeton University in 1976 and the M.A. degree in applied mathematics from Indiana University, Bloomington, in 1979. She won Outstanding Paper Award at the 11th National Computer Security Conference in 1988 and Best Paper Award at the 1987 IEEE Symposium on Security and Privacy. She is founding editor and principal contributor to the Data Security Letter. She is currently serving as the program co-chair for the 1991 IEEE Symposium on Security and Privacy.

- **Catherine Meadows** received the B.A. degree in mathematics from the University of Chicago in 1975 and the Ph.D. degree in mathematics from the University of Illinois in 1981. From 1981 to 1985 she was an assistant professor of mathematics at Texas A&M University. Since 1985 she has been employed at the Naval Research Laboratory, currently in the Center for Secure Information Technology. Her research interests include database security, verification of cryptographic protocols, and executable specifications for secure systems. She has published numerous technical papers on her research. She has served on the program committee for the IEEE Symposium on Security and Privacy from 1987 onwards.

- **Jonathan Millen** received his Ph.D. in Mathematics from Rensselaer Polytechnic Institute, Troy, N.Y., in 1969. His Bachelor's degree is from Harvard University in 1963 and his M.S. from Stanford University in 1965, both in Mathematics. He has worked at The MITRE Corporation in Bedford, MA since 1969. In 1988 he became Principal Scientist in the Distributed Processing Systems Division. He has worked in the area of computer security since 1974, with a special interest in formal methods and analysis tools. He established the IEEE Computer Security Foundations Workshop in 1988 and served as the general and program chair for the 1988 and 1989 workshops. He has also served as the program co-chair for the IEEE Symposium on Security and Privacy in 1984 and 1985. His current activities include modeling a variety of security policies and analysis of covert channels.

MULTILEVEL SECURE OBJECT-ORIENTED DATABASE MODEL
(Panel Position Paper)

Sushil Jajodia

Department of Information Systems and Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030-4444

During the past several years, object-oriented approach to programming and designing complex software systems has received a great deal of attention in the programming languages, artificial intelligence, and database disciplines. There are several aspects of object-oriented model which are quite appealing from security standpoint, and recently, researchers have begun to examine the object-oriented paradigm in the context of security. Several proposals have appeared in the literature dealing with security models for object-oriented databases. While some of them are of considerable interest and merit, they seem to lack in intuitive appeal because they do not appear to model security in a way that would take full advantage of the object-oriented paradigm.

Most security models today are based on the traditional Bell-LaPadula paradigm. While this paradigm has proven to be quite effective for modeling security in operating systems as well as relational databases, it appears somewhat forced when applied to object-oriented systems. The problem is that the notion of object in the object-oriented data model does not correspond to the Bell-LaPadula notion of object. The former combines the properties of a passive information repository, represented by attributes and their values, with the properties of an active agent, represented by methods and their invocations. Thus, the object of the object-oriented data model can be thought of as the object and the subject of the Bell-LaPadula paradigm fused into one.

It seems natural, therefore, to view the system as consisting of objects (in the new, object-oriented sense), where these objects are units of security. Perhaps the most important consequence of adopting such a view is that information flow in this context has a very concrete and natural embodiment in the form of messages. Moreover, taking into account encapsulation,[†] a cardinal property of object-oriented systems, messages can be considered the only instrument of information flow. Information flow becomes explicit (in the form of message exchange among objects) and, therefore, easy to control.

The main elements of the proposed model can now be sketched out as follows. The system consists of objects that are assigned unique classifications. Objects can

[†] Encapsulation of objects—a fundamental property of object-oriented systems—means that only objects themselves can have direct access to their internal state (their attributes). For anyone else to access the object's state, it is necessary to send a message to that object.

communicate (and exchange information) only by means of sending messages among themselves. However, messages are not allowed to flow directly from one object to another. Instead every message is intercepted by the message filter, a system element charged with implementing security policies. The message filter decides how to handle any given message based on the security classifications of the sender and the intended receiver, as well as some additional information. The rulings issued by this module embody the security policy.

The main advantages of the proposed model are its compatibility with the object-oriented data model and the simplicity and conceptual clarity with which security policies can be stated and enforced. Moreover, even though all objects are single-level (in the sense of having a unique classification assigned to the entire object and not assigning any classifications to individual attributes or methods), this does *not* preclude the possibility of modeling multilevel entities. We refer the reader to [1] for additional details.

References

1. Sushil Jajodia and Boris Kogan, ''Integrating an object-oriented data model with multilevel security,'' *IEEE Symp. on Research in Security and Privacy*, pp. 76-85, May 1990.

Object-Oriented System Security

Teresa F. Lunt
Program Manager
Secure Systems

Computer Science Laboratory
SRI International
Menlo Park, California 94025

We are developing a security model for knowledge-based systems for Rome Air Development Center (RADC).[1] Because our initial investigations have shown that knowledge-based systems can be straightforwardly modeled and implemented on object-oriented systems, the first part of this study focused on developing a security model for an object-oriented database system.

Multilevel Object Model

Our first model of a secure object-oriented database system was developed under the SeaView project for RADC. In this model, classifications can be associated with objects (including classes) and *facets* of objects, where a facet could be an instance variable, a method, or a constraint. Thus, in this model, objects can be multilevel; that is, various portions of an object, such as class name, instance variable names, and methods, can have different classifications.

We defined a *hierarchy property*, which requires that the security level of an object must dominate that of its parent class object. This property is needed to permit the object to inherit methods and variables from its parent. This is a fundamental property that will play a part in any model of security for object-oriented systems.

This model also requires that an object's facets be classified at least as high as the object itself. This is analogous to the SeaView property that requires that any data contained in a tuple be classified at least as high as

[1] RADC is funding SRI through subcontract R009406 with IITRI (U.S. Government contract F30602-87-D-0094).

the tuple's primary key, and that any data that can be stored in a relation be classified at least as high as the name of the relation. We called this the *facet property* in this multilevel object model.

Single-Level Object Model

In the current project, we developed a security model for an object-oriented database system in which objects are single-level rather than multilevel. The reasons for doing this were two-fold: first, we uncovered several difficulties with multilevel objects and thus felt that we needed a model for single-level objects to put us on a more solid footing, and secondly, we felt that a single-level object model might be sufficient to support a multilevel knowledge-based system.

This single-level object model includes the hierarchy property, discussed above. We demonstrated that typical database security and integrity policies can be supported by this model.

Composite Object Model

In our ongoing work, we are again investigating a model that supports multilevel objects. Our investigations of actual knowledge-based system requirements have led us to the conclusion that multilevel objects are desirable to support a knowledge-based system. We have also developed a new model that does not have the difficulties of our initial multilevel object model discussed above. In essence, this new model combines the features of the above two models. In this new model, an object can be multilevel, but the access class of the object dominates the access class of its components (as contrasted with the facet property described above). Each of these components is itself an object, which in turn can be multilevel and can consist of lower-level components. This provides a very natural and simple way of constructing multilevel objects. This model also leads to a natural decomposition of multilevel objects from single-level partitions. Thus, queries and updates on multilevel objects will be decomposed into queries and updates on the single-level partitions. This approach is easily generalized to an object model in which the object base is distributed and queries and updates are distributed across numerous processors and memories.

Discretionary Access Controls for Object-Oriented Database Systems

In another project for RADC[2], we are developing a model for discretionary access control in object-oriented database systems. This model allows the implementation of arbitrary access policies and enables one to easily implement such conventional mechanisms as access control lists, named access control lists, user groups, user attributes, user capability lists, and user roles. It also provides convenient extensions for additional access restrictions based on time, day, date, or any user-supplied function. The generality of the mechanism permits users to think of their access policies as true policies rather than as particular representations imposed by more restrictive mechanisms.

We added the notions of *negative authorizations* and *strong* and *weak* authorizations, so that we could require more and more privilege in order to access objects as you go down the class hierarchy. Negative authorizations are used to express explicit denial of authorization, that is, those users and groups that must be denied authorization even if authorization is later granted. Strong authorizations are authorizations that cannot be overridden. Weak authorizations can be overridden by strong authorizations. Weak and strong positive and negative authorizations can be flexibly combined to make realistic policies.

[2] RADC is funding SRI through subcontract C/UB-07 with CALSPAN, U.S. Government contract number F30602-88-D-0026

Questions in Trusted Object-Oriented Database Management Design

Catherine Meadows
Code 5543
Center for Secure Information Technology
Naval Research Laboratory
Washington, DC 20375

A few years ago, Carl Landwehr and I wrote a paper [2] in which we showed how a security model could be developed and described from an object-oriented point of view. In it, we outlined the model for the NRL Secure Military Message System [1], and restated it in an object-oriented framework, in terms of objects that could use certain methods and pass certain messages between each other. We found certain aspects of the object-oriented approach very useful in describing and developing the security model, and we suggested ways in which an object-oriented system could be designed so that it could be used to enforce such a model.

We found the concept of class hierarchies and inheritance among classes extremely useful in developing the object-oriented SMMS model. A security policy could be defined for one class of objects which could then be inherited by other classes. This policy could then be refined for application to these other classes. Thus, for example, an object of class container would obey the security rules relevant to containers (for example, the rule that a container cannot contain any entities whose labels are not dominated by its container label), while a container of class message would obey, not only the container rules, but the rules relevant to messages (for example, the rule that a message marked 'sent' may not be modified). Such an approach provided a straightforward and understandable way of describing a security model.

Another concept that we found useful was that of composite object. This allowed us to model policies for handling objects that were made up of several different objects that each obeyed their own policies, and that possibly existed at different security levels. Thus, for example, we could describe policies governing message files made up of messages at different levels, and policies governing messages made up of headers and paragraphs of different levels.

We never got to the point of designing an object-oriented system that would assist us in enforcing such policies. But we did identify several properties that such a system must have in order for it to be useful. In particular, it was necessary that it be possible to ensure that an object responded only to those messages to which it was supposed to respond and used only the appropriate messages in its response, that inheritance among object classes be strictly enforced, and that messages and methods defined further down in the class hierarchy did not interfere with security-related methods and messages defined earlier. It was also necessary, although we did not explicitly say so in our paper, that the system be able to support multilevel composite objects and enforce security policies defined for such objects.

With these thoughts in mind, I would like to turn to the other members of this panel and ask how their systems would enforce such policies. In particular, how do their systems enforce inheritance, and do they think it possible that the inheritance mechanisms of their systems could be used to define and enforce security policies? Also, do their systems support security policies for composite objects and multilevel composite objects, and in what way? Finally, do they think that such features are important or necessary? If not, what features do they think are important?

References

1. C. E. Landwehr, C. L. Heitmeyer, and J. McLean, "A Security Model for Military Message Systems," *ACM Transations on Computer Systems*, vol. 2, no. 3, pp. 198-222, August 1984.

2. C. A. Meadows and C. E. Landwehr, "Designing a Trusted Application Using an Object-Oriented Data Model," in *Research Directions in Database Security*, ed. T. F. Lunt, Springer-Verlag, to appear.

Single-Level Objects for Security Kernel Implementation

Jonathan K. Millen
The MITRE Corporation
Bedford, MA 01730

The design for a secure knowledge-based system given in [1] takes an object-oriented approach. A knowledge-based system is a database system with certain additional features, such as a class hierarchy with inheritance, and support for rule-based inference. The class hierarchy comes naturally with an object-oriented approach. As for the inference engine, we found that it can be added within the existing structure without straining the system organization or security properties.

To obtain the greatest possible assurance for mandatory security, we took a layered approach to the system design, with the idea of building on a conventional security kernel. It was assumed that this kernel would enforce mandatory security while providing essential services such as memory management, process management, input-output, and user identification and authentication. We investigated the feasibility of superimposing on this kernel an object-oriented layer, to provide primitive services such as object creation, message handling, inheritance, and object data accesses.

It was an important objective to determine whether the object layer could operate within the security constraints imposed by the kernel, without any privilege to violate those constraints, and yet provide a natural, understandable and usable interface. We found a suitable higher-level, object-oriented security policy based on the simple notion that each object has a single security level.

The security level of an object applies to everything in the object, whether the object represents a class or an instance. Attributes of a class object, such as the methods that belong to it, the names and types of its variables, and any default values they possess, are all labelled at the object level. Attributes of an instance object, namely the values of its variables, are labelled by its level. In fact, the design and policy do not make any distinction between classes and instances; they are all objects, and the policy applies to them all uniformly.

Saying that each object has a single security level (applying to everything in it) is not by itself a full or unique description of a security policy. A number of other design choices were made and details were filled in. For example, not all "objects" in the widest sense are labelled - only named, uniquely identifiable objects that serve as containers for data values. Instances of data values - numbers, strings, and other entities of types normally predefined in the system - are stored inside labelled objects.

When the value of a variable in one object is a reference to another labelled object, the reference itself, stored inside the first object, is distinguished from the object it references. The former is implicitly labelled with the label of the object containing it. Consequently, it is possible for the existence of an object to be known (through the reference) at a lower security level than the level of the object.

An important design choice was made in the handling of messages. Any object can send a message to any other object. A "subject" or method invocation is created upon reception of a message. The subject is given a security level that dominates both that of the receiving object and that of the message (which is the level of the subject that sent it). In some cases, the new subject level will strictly dominate that of the receiving object, and the subject will then not be allowed to update variables in that object.

Another aspect of the security policy was necessitated by the mechanism of inheritance. Since a message might be handled using a method inherited from a parent class, the parent class of an object should be at a lower or equal security level. This "hierarchy" property is reminiscent of the directory hierarchy compatibility property of secure Multics.

With regard to the "trustedness" of the object layer, we are assuming that the pre-existing security kernel will enforce the mandatory policy at its interface without granting any privileges to the object layer, or expecting any guarantees. This means that we can use an off-the-shelf evaluated conventional secure operating system. However, an off-the-shelf non-secure object-oriented environment would require substantial redesign to respect the kernel security constraints, specifically to support the policy we have outlined. The object layer may also introduce and be trusted to enforce its own discretionary policy, which may or may not make use of a discretionary access control mechanism present in the kernel.

We feel that the single-level-per-object design is appropriate for users of the object system interface. Although multilevel objects could conceivably be supported using a decomposition approach similar to the way multilevel relations are handled in SeaView, there are new complications in an object-oriented environment due to the class hierarchy. Furthermore, the single-level approach avoids polyinstantiation and clarifies integrity issues.

[1] J. K. Millen and T. F. Lunt, "Security for Knowledge-Based Systems," MTR-10686, The MITRE Corporation, September 1989, and SRI-CSL-90-04, SRI International, August, 1989.

ISSUES IN MULTILEVEL SECURE OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEMS - A POSITION PAPER

Bhavani Thuraisingham

The MITRE Corporation, Burlington Road, Bedford, MA 01730

1. INTRODUCTION

Object-oriented systems are gaining increasing popularity due to their inherent ability to represent conceptual entities as objects; this is similar to the way humans view the world. This power of representation has led to the development of new generation applications such as CAD/CAM, Multimedia information processing, Artificial Intelligence and Process control systems. However the increasing popularity of object-oriented database management systems should not obscure the need to also maintain security of operation. That is, it is important that such systems operate securely in order to overcome any malicious corruption of data as well as to prohibit unauthorized access to and use of classified data especially with military applications.

Recent investigations of security issues in object-oriented systems has proceeded in three directions. In the first direction, multilevel object-oriented data models are being developed in order to represent a database with data at different sensitivity levels. Thus multilevel secure object-oriented database management systems (MLS/ODBMS) will be needed to manage a multilevel object-oriented database. In the second direction, an object-orient approach is taken to design a secure software system. That is, the software system is treated as a set of interacting objects. In the third direction, an object-oriented approach is taken to design trusted applications. In this paper we focus only on the issues for designing a MLS/ODBMS. In particular, the issues on data modelling, architectures, interface languages, metadata management, query processing, transaction management, and theory, will be discussed.

2. MULTILEVEL OBJECT-ORIENTED DATA MODEL

A multilevel object-oriented data model should support all of the constructs associated with an object-oriented data model. In this section we will briefly identify the various constructs of an object-oriented data model and discuss the security issues.

All conceptual entities in an object-oriented database are modelled as objects. A group of objects with similar properties form a class which is also an object. The objects that form the class are called the instances of the class. A class could be a system-defined class, such as a class of integers or strings, or it could be a user-defined class such as a class of documents. A class has methods associated with it. Methods provide a means for accessing the class or the instances of the class. Methods may be executed by sending messages to the class or to its instances. This feature is known as encapsulation.

Associated with each class is a set of instance variables that describes the attributes of the instances of the class. An instance variable could be either a noncomposite instance variable, or it could be a composite variable. Noncomposite instance variables are divided further into simple instance variables and complex instance variables. Simple instance variables take individual objects as their values. An individual object could be a basic object such as an integer, string, or boolean, or it could be an object which is described by a tuple value such as an employee, a department, or an automobile. For example, the simple instance variables of a book class include the "Author" and the "Title." Complex instance variables take a set or a list of individual objects as their values. Examples of complex objects include a set of names or a set of employees. Composite instance variables describe the components of the instances. That is, a component of an instance is part of the instance. For example, a document could be composed of a cover, table-of-contents, set of sections, and references.

Any class which has a composite instance variable is a composite class. The instances belonging to such a class are composite objects. A composite object together with its components forms a hierarchy called the IS-PART-OF hierarchy (or composite hierarchy). The link from a composite object to its component is called a composite link. A second hierarchy that may be formed is the IS-A hierarchy, where subclasses are associated with a class. The subclasses inherit all the methods and instance variables defined for a class. This feature is known as inheritance. A subclass could also have some additional instance variables and methods. Other features that need to be supported include versioning. That is, different versions of an object need to be maintained. These versions could be alternate versions or historical versions that evolve over time.

In order to develop a multilevel object-oriented data model, security properties between the various constructs need to be enforced. One of the main issues that need to be resolved is whether an object should be single level or multilevel. If an object is single level, then all its components and parts are assigned the same security level. If an object is multilevel, then its components could have different security levels. It appears that at the conceptual level the notion of multilevel objects is desirable in order to reflect the real-world more accurately However, the multilevel objects could be decomposed into single level objects and stored in single level files or segments in order to obtain higher levels of assurance. A multilevel object-oriented model should also support polyinstantiated versions. Various definitions of polyinstantiation have been proposed. We assume that polyinstantiation occurs when users at different security levels have different views of the same entity. Cover stories at different security levels can be regarded as a form of polyinstantiation.

3. ARCHITECTURURAL ISSUES

The first step towards the design of any software system is to evaluate a variety of architectures and then select the most appropriate ones for the design and implementation of the system. In this section, we describe four possible architectures that have to be evaluated with respect to the various functions of a MLS/ODBMS and the level of assurance expected of the system.

In the first of these architectures, an object manager is hosted on top of a multilevel relational system. The object manager is responsible for managing complex multilevel structures, consequently alleviating the burden placed on the application programs. The multilevel relational system acts as the storage manager and is responsible for concurrency control and recovery. Such an architecture would be very useful in the near-term. This approach takes advantage of the developments in multilevel relational systems.

In the second of these architectures, a MLS/ODBMS is hosted on a trusted computing base (TCB). The object manager as well as the storage manager are part of the MLS/ODBMS. Such an architecture is preferable to the first. However, several issues need to be investigated before this architecture can be developed.

In the third of these architectures, a trusted front-end machine is connected to untrusted back-end Object-oriented DBMSs. Each back-end machine operates at a single level and manages the data at the same level. Recently multilevel relational systems have been developed using the trusted front-end / untrusted back-end approach. The feasibility of developing a MLS/ODBMS using such an approach needs to be investigated.

In the fourth of these architectures, a trusted filter needs to be placed between the untrusted front-end user interface and the untrusted back-end Object-oriented DBMS. This is based on the integrity lock architecture. The feasibility of developing a MLS/ODBMS using such an approach needs to be investigated.

The MLS/ODBMS design will no doubt depend on the architecture that is selected. From a preliminary evaluation of the various architectures, the second architecture appears to be the most appropriate one for high assurance systems. Furthermore, by having the object manager and the storage manager together as part of the MLS/ODBMS will result in better performance. The security issues that we have identified in this paper are related to the second architecture.

4. LANGUAGE ISSUES

Data manipulation, data definition, and persistent database programming languages are being developed for Object-oriented DBMSs. Data manipulation languages provide the facility for interactively querying the database. Data definition languages provide the facility for defining the various types of data. Persistent database programming languages that have been developed for object-oriented systems support many of the features of modern object-oriented languages such as abstract data types, as well as support persistence where data can be accessed even after the process that creates it is not in existence.

Although object-oriented database languages have received considerable attention, extensions of these languages to support multilevel security has not been done. Note that extensions to relational database languages to support multilevel security assertions has been reported in the literature. Therefore it is proposed that for object-oriented database languages the constructs for multilevel security should be incorporated at the onset.

5. METADATA MANAGEMENT

Metadata management in an Object-oriented DBMS includes the creation, modification and deletion of the various classes, security constraints which assign security levels to the data, integrity constraints which enforce consistency among the data, and relationships between the various constructs associated with the object-oriented data model. Metadata is managed by the metadata manager component of the DBMS. The metadata required by the other modules such as the query processor and transaction manager are accessed via the metadata manager. The metadata could be static, in which case it is assumed that all of the metadata are created before the database is created, or the metadata could be dynamic, in which case it is assumed that the metadata is evolving.

The security impact on metadata management has not received much attention even in multilevel relational systems. Only a few of the designs have addressed the issues involved to some extent. Therefore much research needs to be done in the area of metadata management in multilevel systems. One function of the metadata manager in a MLS/ODBMS could be ensure that the classes are created at the appropriate security levels. That is, the security constraints could be applied when the metadata is created and it could be ensured that the constraints are satisfied. This approach is not desirable for applications with dynamic security constraints. The metadata manager is also responsible for ensuring that the metadata is consistent. For this purpose tools for checking the consistency of the metadata need to be developed. In a multilevel environment it has to be ensured that the security constraints are enforced in such a way that security violations cannot occur. For example, the following constraints "John lives in Boston is Unclassified", "The existence of a man named John is Secret" are inconsistent with one another. The metadata manager should be able to detect such inconsistencies. In the case of distributed data, the metadata manager is responsible for maintaining the information on data distribution.

It should be noted that the metadata itself could be assigned different security levels. For example, the metadata could be maintained either at system-low, system-high, or at different security levels. Some implications of these configurations for multilevel relational systems have been investigated. They need to be investigated for a MLS/ODBMS also.

6. QUERY PROCESSING

The most important function of a MLS/ODBMS is to provide a facility for users to query the database system and obtain authorized and correct responses to the queries. The data is usually distributed in files or segments at different security levels. The distribution of the data across security levels should be transparent to the user. That is, the user should query the database and receive only the responses at or below his level. The query processor, which is responsible for handling queries, should determine the segments which store the relevant data and assemble the data from the different segments.

Following are some of the issues in secure query processing.

(1) Access methods: Two of the major components of the system architecture are the object manager and the storage manager. The object manager is responsible for managing the complex objects and the storage manager manages the multilevel database. Efficient and secure access methods for the object and storage managers need to be developed.
(2) Query Optimization: Special query optimization strategies need to be developed to access the classes, objects and instance variables of objects. The security impact on query optimization techniques need to be investigated.,
(3) Polyinstantiation: In an object-oriented data model, the instances, instance variables and even the classes could be polyinstantiated. Strategies for efficiently handling polyinstantiation should be developed.
(4) Data Distribution: The data could be distributed in files or segments at different security levels. Strategies for recombining the data at different security levels during query processing need to be developed.
(5) Inference Problem: Inference is the process of forming conclusions for premises. This process becomes a problem when unauthorized conclusions are drawn from authorized premises. Techniques for handling the inference problem in multilevel relational systems are being investigated. These techniques need to be adapted for a MLS/ODBMS.

7. TRANSACTION MANAGEMENT

Transactions are managed by the Transaction Manager component of a database system. A transaction is a program unit that must execute in its entirety or not execute at all. The issues involved in transaction management are concurrency control and recovery. Concurrency control techniques ensure the consistency of the database when multiple transactions execute concurrently. Recovery techniques are necessary in order to ensure that the database is brought to a consistent state when transactions are aborted due to some failure. Although much progress has been made in incorporating transaction management features into a DBMS, providing these features in an Object-oriented DBMS as well as in a multilevel relational DBMS has only recently begun. Very little attention has been given to transaction management in a MLS/ODBMS.

Following are some of the issues on transaction management in multilevel database systems.

(1) Security policy: The security policy enforced by the system needs to be extended for transaction management. An important issue that needs to be addressed is whether a transaction operates at a single level during its execution or whether it can change security levels during execution. If it is the former, then the transactions are single-level. If it is the latter, then the transactions are multilevel.
(2) Serializability condition: Appropriate serializability conditions for transaction management have to be formulated. These conditions should ensure that only serializable schedules are executed.
(3 Concurrency control: The various concurrency control techniques need to be examined for transactions which operate in a multilevel environment. These techniques should ensure that not only is the consistency maintained, but the actions of higher level transactions do not interfere with lower level ones. In addition, deadlocks and starvations have to be prevented. The three techniques that are being investigated for multilevel relational systems are locking, timestamping and validation. These techniques need to be examined for a MLS/ODBMS. In addition, concurrency control techniques based on abstract data types that are being investigated for an Object-oriented DBMS need to be adapted for a MLS/ODBMS.
(4) Recovery: Recovery techniques should ensure that the system should be recovered to a consistent state in the event of any failure. Recovery techniques that are being proposed for an Object-oriented DBMS need to be adapted for a MLS/ODBMS.

8. THEORY

Secure object-oriented database systems could be built by incorporating multilevel security into existing Object-oriented DBMSs by the use of special annotations and supporting mechanisms. Such an ad-hoc approach is usually cumbersome and awkward especially because it involves the cooperation of different and disparate expertise including those of data model designers, systems designers and language designers. Therefore, a simple but adequate model for the description and construction of a multilevel secure object-oriented database management system is certainly desirable. That is, we need a theory for integrating the data modelling, programming language and system development concepts to aid in the construction of multilevel secure object-oriented database management systems.

During the 1970s the notion of using type theories for representing Abstract Data Types in programming languages was proposed. Abstract Data Types are also an important concept in the object-oriented paradigm, and type theories have subsequently been used to provide a theoretical framework for object-oriented systems. The inheritance mechanism, another important concept in the object-oriented paradigm, has also been supported in certain type theories. Among the enhancements that have resulted from such work, programming language compilation and execution have been made more efficient and support for persistent objects essential to database applications has also been provided

It is important to note that type theories that have been developed for the object-oriented paradigm in general are not sufficient in themselves to achieve the goals of constructing multilevel object-oriented systems. This is because the existing type theories do not have security levels incorporated into them and therefore cannot be used for computations in a multilevel environment. What is needed is a type theory developed specifically to meet the needs of a multilevel operating environment.

Work should be directed towards developing a multilevel type theory from first principles with the necessary extensions to be made consistently and systematically for various tasks such as data modelling, programming language design and system construction. It appears that our basic intuitions concerning types in a multilevel environment could be represented and formalized using ideas from universal algebra and second order models, such as those based on lambda calculus and constructive mathematics.

9. OTHER ISSUES

In this section, we discuss some of other issues that have to be addressed. They are (1) designing an Intelligent MLS/ODBMS, (2) steps to developing a MLS/ODBMS, and (3) designing a Distributed MLS/ODBMS.

An Intelligent MLS/ODBMS should have the capability to make deductions and intelligent decisions. At present, Object-oriented DBMSs are being extended with rule processing and deduction capabilities. The security impact of these extensions are also being investigated. Such work will produce database systems that are not only multilevel secure, but also have powerful representational and deduction capabilities.

The steps required to develop a MLS/ODBMS depend on the level of assurance that is expected of the system. In addition to the design and implementation steps (which will consist of the design and implementation of the models for query processing, update processing, transaction management, and metadata management), a MLS/ODBMS development process would include the following: (1) Security Policy: This is a discussion on the policies for mandatory security, discretionary security, integrity, auditing, authentication, accounting, and the roles of the database administrator, and the systems security officer. (2) Model: A security model for the MLS/ODBMS should be developed. (3) Specification: Security requirements expressed by the model are specified (either formally or informally), and the correspondences between the specification and the model and the specification and the implementation are demonstrated.

The issues in designing a Distributed MLS/ODBMS will include all of the issues involved in designing a MLS/ODBMS. Furthermore, the additional complexities that arise from distributing the data across several nodes should also be handled. Data distribution causes a major impact on the functions of query processing, transaction management and metadata management. In an object-oriented data model, data distribution causes additional problems if encapsulation and inheritance properties have to be preserved. The security impact on the various functions and properties remains to be investigated.

10. CONCLUSION

In this paper we have described the need for a MLS/ODBMS and identified some security issues that need to be given further consideration. Various efforts are under way to provide solutions to some of these issues. More research, prototype development, and simulation studies need to be carried out before useful MLS/ODBMSs can be developed successfully.

C2 SECURITY AND MICROCOMPUTERS

Angel L. Rivera
Sector Technology
5109 Leesburg Pike
Suite 900
Falls Church, Virginia 22041-3201
(703)845-5600

The latest and most commonly used marketing (claim-plot) term being used by manufacturers of PC security products is that their products meet C2 functionality, a term that has never been explained or described by the folks at NCSC.

The thinking behind such a plot is that if the product has discretionary access controls (DAC), user identification and authentication (I&A), audit trails (AUD), it more or less meets the C2 requirements. The fact of the matter is that no PC security product meets C2 requirements because they are subsystems, and in order for them to get a C2 rating they would have to be a complete ADP system that has been evaluated and certified by the National Computer Security Center (NCSC). Will we ever have a C2 PC? Your guess is as good as mine.

What I would like to accomplish in this article is to clarify the requirements for D2 evaluation and give a plain english definition of the features needed to meet the D2 requirements, which is a different interpretation of C2 for subsystems. I will also include a matrix for you to use when evaluating PC security products under consideration. In my opinion, D2 is the closest we will ever get to C2 in a PC.

As you can imagine, PC security product manufacturers were pretty upset that the NCSC would only evaluate their products and would not rate them. In response to the demand from users and the vendors, NCSC came up with a rating system at the D level. For an explanation of these ratings, I suggest you read the booklet titled "COMPUTER SECURITY SUBSYSTEM INTERPRETATION of the Trusted Computer System Evaluation Criteria" (NCSC-TG-009)[1] . Since subsystems, by their very nature, do not meet all of the requirements for a class C1 or higher computer system, it is most appropriate to associate subsystem ratings with the D division of the TCSEC. This Interpretation defines the D1, D2 and D3 classes within the D division for subsystems. The D1 class is assigned to subsystems that meet the interpretations for

[1] For the purpose of this discussion, I will call this book the Light Blue Book.

requirements drawn from the C1 TCSEC class. Likewise, the D2 class consists of requirements and interpretations that are drawn from the C2 TCSEC class. The D3 subsystem class is reserved for DAC subsystems and audit subsystems that meet the B3 functionality requirements for those functions. A plain D rating means that the subsystem does not meet all of the requirements for a higher category[2] .

Furthermore, subsystems are evaluated on the following subsets: Discretionary Access Control (DAC), Object Reuse (OR), Identification and Authentication (I&A), and Audit (AUD). Each subset has its own set of ratings as identified in the following table:

SUBSYSTEM FUNCTION	**POSSIBLE RATINGS**
Discretionary Access Control	DAC/D DAC/D1 DAC/D2 DAC/D3
Object Reuse	OR/D OR/D2
Identification & Authentication	I&A/D I&A/D1 I&A/D2
Audit	AUD/D AUD/D2 AUD/D3

When subsystems are evaluated for more than one function, they will receive a separate rating for each function.

Back to our topic of C2 functionality. The closest one can get to a C2 rating with a PC security product, is to have a product that meets at least the D2 requirements in all of the functions. It would be nice if we could go to the Evaluated Products List and pick the D2 certified products, however, there are none as of yet, and it takes anywhere between six months to a year to get a product evaluated by the center. In the meantime, unfortunately, the evaluating is left to us, the users. I will try to walk you through the D2 requirements and then give you a matrix for your own use in evaluating available PC security products.

[2] NCSC-TG-009

Discretionary Access Controls (DAC)

Let's start with discretionary access controls (DAC). The purpose of the DAC subsystem is to control access to resources. To do this, the security package must use some mechanisms to determine whether users are authorized for each access attempted. For example - user one wants to run a word processing program. The DAC mechanism must identify this user and determine whether he has access to that program or not. If the user decides to open a file with the word processing program, the DAC mechanism should also determine if the user has access to the file being opened.
Under the D2 requirements this mechanism shall allow users to specify and control sharing of those files (objects) by named individuals, or groups of individuals. In other words, the DAC mechanism should allow the authorized user to share the files that belong to him.

This sharing of files include individual or groups of individuals. The security package should be able to limit the ability to grant the propagation of access to only authorized individuals. Some users could be allowed to share some of their files while others could be restricted and not allowed to share files.

In addition the security package should have a default setting that denies all users access to files on programs when no explicit action has been taken by the authorized user to allow access. For example: if I create a file the default permissions on that file are no access for other users unless I specify otherwise.

According to the Light Blue book (NCSC-TG-009), mechanisms that can meet the DAC requirements include, but are not limited to: access control lists, capabilities, descriptions, user profiles, and protection bits.

Object Reuse (OR)

In plain english, object reuse means that before I let you use a storage medium like a diskette, I have to make sure that the diskette has no data. Erasing a diskette, as most of you know, does not get rid of the data, it only gets rid of the pointers to that data and frees the space for reuse. With common disk utilities, the data could be recovered. The same applies to memory. When a word processing package loads a file into memory for you to work on, and then saves it to disk when you are finished, the work space used in memory is not erased. It is like the diskette, it is made available for reuse. A dump of the contents of memory would display the contents of that file. In order for a PC security product to meet this requirement, the package would have to overwrite, with meaningless or unintelligible bit patterns the files that it erases, also known as file purging, and overwrite the contents of memory, also known as memory purging. Keep in mind that this requirement also applies to hard disks, tapes, i/O buffers, temporary files, CMDS memory, printer and keyboard buffers, numerous registers, etc.

Under the D2 requirements, even encrypted data needs to be overwritten. The reasoning behind this is that an unauthorized user can get this encrypted data and try to figure out the key. According to the Light Blue book, there is an alternative way of approaching the problem of object reuse and that is to deny read access to the previously used storage objects until the user who has just acquired them has overwritten them with his own data. One way of doing this is to have a security package that does not allow unauthorized users to do memory dumps or do a file unerase, and does not allow sector reads. Unfortunately, software only security packages cannot take advantage of this alternative because any user can load the Operating System from the A drive and do absolute sector reads with common disk utility packages.

Identification and Authentication (I&A)

The simplest way to meet this requirement is through the use of a user ID and a password. More sophisticated methods for identification and authentication include, but are not limited to smartcards, fingerprints, tokens, etc. Keep in mind that the claimed identification of a user must be authenticated by an explicit action of the user. I&A must be a two step process. The user ID can be public, however, by keeping it secret, it can only strengthen the I&A portion of the security system. The password on the other hand must be secret. The authentication data (i.e. passwords) must be protected so that it cannot be accessed by unauthorized users. Here again, software security falls short of the requirement. Even if the password file is encrypted, it must be protected from unauthorized access, and software by itself cannot prevent this from happening.

Besides being able to enforce individual accountability, the security system must be able to associate the user ID with all auditable actions taken by that individual. For PCs, auditable actions include the date and time of the event, the user ID, type of event (i.e. file opened, file deleted, password changed, etc.), and the success or failure of the event.

Audit (AUD)

The purpose of the audit subsystem is to record all the security-relevant actions that take place on the PC. This will allow the administrator to detect security violations and trace them back to the responsible parties.

On mainframe systems it would be possible to use the I&A mechanisms and DAC mechanisms built into the operating system and pass this data to the Audit subsystem for storage. However, on PCs there are no built in I&A or DAC mechanisms. Therefore, the audit subsystem must be used in conjunction with a DAC and I&A. This does not mean they have to be in the same security package. For example, you could use a biometric device (like a finger print reader) as the I&A mechanism and then pass the I&A data onto the Audit subsystem. This example is more expensive to implement but more secure.

Under the D2 requirements for Audit, the system should protect the audit trails from unauthorized access, modification, or destruction. Here again, software-only security packages fall short of the requirements.

Software-only packages can protect the confidentiality of the audit data (i.e. if they are encrypted) but cannot protect against unauthorized access, modification, or destruction. Once a user boots the system from the A> drive and uses common disk utility to access the medium storage (where the audit trails are stored) the requirement has been compromised.

Authorized access should be clarified when discussing the Audit requirements. Just because you are the system administrator and have authorized access to the audit trails does not mean you can access them any which way you want. Under the D2 requirements this audit data must be accessed through the audit subsystem. This requirement makes sense since you want to make sure users with authorized access can only access the audit data with the approved audit mechanism and that this action is recorded itself in the audit trails.

Another D2 requirement is that the Audit subsystem allow the system administrator the capability to selectively audit some users while not auditing others. To accomplish this, the subsystem should have the ability to perform selection of audit data based on individual users. There are two ways to select the data. You can either pre-select the events actions that the audit subsystem is going to keep track of, or have the subsystem extract specific data out of a more general audit trail. The main advantage of this requirement is that it reduces the amount of data a system administrator has to review.

The specific information to be recorded about each security-relevant event includes the following:

- Use of I&A Mechanisms
- Introduction of objects into a users address spool (e.g. file open, program run)
- Deletion of objects
- Actions taken by computer operators, and system administrators/security officers

For each recorded event the audit record should include:

- Date and time of event
- User
- Type of event
- Success or failure of event

Keep in mind that this does not mean that every time anyone uses the word processor you need to record it. Remember, we are talking about security-relevant events only. If the word processor is not used to generate sensitive documents you do not need to keep track of who uses it. On the other hand, if it is used for sensitive documents you should record that the program was used and which files were created, modified, read, or deleted.

You have probably realized by now that there are some very strict requirements to meet C2 functionality or get a D2 rating. Not only are there requirements, but there are different categories. When a vendor tells you they meet C2 functionality, your immediate response should be to ask what is C2 functionality. Second, you should ask under which categories. The ideal product would meet the requirements in all the categories. To make sure the vendor knows what he is talking about, ask him/her to name the categories.

To refresh your mind, they are: Identification and Authentication (I&A), Discretionary Access Controls (DAC), Object Reuse (OR) and Audit (AUD).

Requiring C2 functionality for all your microcomputer security needs can be overkill in many areas. The degree of security needed is dependent on the sensitivity of the data. In plain and simple terms, if your data is worth $10 you do not spend $15 to protect it. C2 functionality should only be required in your most sensitive applications. And remember that C2 functional products can not be used to process classified data.

This article only discussed the feature requirements for subsystems. The reader should should be aware that there are also assurance, integrity and documentation requirements that need to be met in order to get a rating. The chart provided is designed to give the reader a quick picture of a subsystem's capability. it is not an all inclusive chart. For more detail requirements consult the NCSC Computer Security Subsystem Interpretation booklet (NCSC-TG-009).

Angel L. Rivera, an internal consultant for Sector Technology is the author of this article. Mr. Rivera is also the President of H & A Micro Consultants. Copyright 1989.

C2 FUNCTIONALITY CHART

	YES	NO
DISCRETIONARY ACCESS CONTROLS		
Controls access to application programs:		
Controls access to individual files:		
Controls the sharing of files by user:		
The default file permissions can be set to no access to other users:		
OBJECT REUSE		
Overwrites files when erased: or Denies read access to previously used storage until its overwritten with new data[1]:		
Overwrites scratch memory: or Denies access to scratch memory until its overwritten with new data:		
IDENTIFICATION & AUTHENTICATION		
Is the I&A at least a two-step process (i.e. user ID, token or password):		
Is the authentication data protected from unauthorized access[1] :		
Can the system associate the user ID with all auditable actions taken by that individual:		

[1] Software only products do not meet this criteria

	YES	NO
AUDIT		
Can the system protect the audit trails from unauthorized access[1]:		
Can the system protect the audit trails from unauthorized modification[1]:		
Can the system protect the audit trails from unauthorized destruction[1]:		
Can the system prevent access to the audit trails without using the audit mechanism (i.e. can the system prevent the administrator from accessing the audit trail without the security system):		
Can the system administrator audit some individuals while not auditing others:		
Does the system have the ability to perform selection of audit data based on individual users:		

[1] Software only products do not meet this criteria

Executive Summary

Electronic Certification: Has Its Time Come?

Miles Smid

National Institute of Standards and Technology

Electronic certification is now being used or considered for use in financial, defense logistics, electronic data interchange, and electronic mail applications. Electronic certification employs a digital signature to detect unauthorized modifications of data and to authenticate the identity of the person generating the signature. In electronic data systems, digital signatures represent the analogue of written signatures on documents. Both secret-key and public-key cryptography may be used to implement an electronic certification system, however, public-key algorithms provide the added benefit of non-repudiation. Panel members discuss the present status of electronic certification and the obstacles which must be overcome in order for it to be more widely implemented throughout the government and commercial sectors.

FUNCTIONAL IMPLEMENTATION OF C2 BY 92 FOR MICROCOMPUTERS

Second Lieutenant Alan R. Berry

Air Force Cryptologic Support Center (AFCSC/SRVC)
San Antonio, Texas 78243-5000

ABSTRACT

This paper will provide some functional insight on how to implement the "C2" level of security on microcomputers. This can quickly become a very complex proposition, so the scope will be limited to stand-alone PCs and use existing Department of Defense (DOD) and Air Force regulations as the basis for the paper. The first argument almost always encountered is that a stand-alone PC usually operates in a dedicated mode, thus excluding it from the "C2 by 92" requirement [5],[6]. So the first step will be to define "dedicated" mode [5], along with a quick look at why security is necessary. Each part of the C2 level security [7] will then be discussed and applied to the microcomputer scenario.

INTRODUCTION

Perhaps the biggest buzzword or phrase in DOD Computer Security circles is "C2 by 92." Exactly how it applies to each user's system varies as much as the spots on a leopard. In fact, some computer systems do not even fall under the regulations requiring C2 [5]. This paper will help to define what systems need the C2 level security and give some functional ideas and examples on how to implement microcomputer security. I say functional ideas because most PCs do not lend themselves to automated security and yet we must provide security in some way. To put this into perspective, automated computer security is only one of six parts for providing security on our computer systems. The other five are procedural controls, personnel security, physical security, COMSEC, and TEMPEST. To provide a functional C2 level of trust, we can also increase security in one or more of the other areas.

THE DRIVING FORCE

DODD 5200.28, Security Requirements for Automated Information Systems (AISs) [5], drives all existing DOD requirements in Computer Security. It states:

"All AISs that process or handle classified and/or sensitive unclassified information and that require at least controlled access protection (i.e., class C2 security), based on the risk assessment procedure described in enclosure 4, shall implement required security features by 1992."

Enclosure 4 contains a table listing systems running in the dedicated mode as having no minimum security class, but notes that most dedicated systems warrant at least C1 protection. This leaves the obvious question of what is dedicated mode in the microcomputer world.

DEDICATED MODE

The National Computer Security Center (NCSC) defines dedicated mode as:

"....when each user with direct or indirect individual access to the AIS, its peripherals, remote terminals, or remote hosts, has all of the following:

a. A valid personnel clearance for all information on the system.

b. Formal access approval for, and has signed nondisclosure agreements for, all the information stored and/or processed (including all compartments, subcompartments and/or special access programs).

c. A valid need-to-know for all information contained within the system" [9].

A working definition for microcomputers operating in dedicated mode can then be as follows:

Any stand-alone microcomputer operating in an area where all personnel with physical access to the system have the clearance and need to know for all data on the system.

Who qualifies for dedicated mode and therefore earns the exception to "C2 by 92?" Just about everybody. The decision remains with the user and must be made on a case-by-case basis. One of the biggest problems with security is over-zealousness. Do not go overboard and excessively protect your word processing workstations if they fit the dedicated mode of operation. But on the other hand, do not be afraid to use some security techniques even on a dedicated system. Perhaps your particular system does not need all of the C2 characteristics like object reuse, but just the Discretionary Access Control (DAC), Identification & Authentication (I&A), and audit trails [7]. The remainder of this paper will provide some functional techniques for all users to consider, regardless of their mode of operation and security requirements.

WHAT IS C2?

The best answer for that question is found in DOD 5200.28-STD [7], the "Orange Book." It stresses 2 types of requirements, specific security features and assurances. A feature is the mechanism that provides the functionality, and an assurance is a degree of trust that the feature will work. Each C2 requirement is discussed below with several options available for most sections, giving the user a choice. No single answer is universal, and in the end the user must decide what works for his individual system.

Instead of repeating everything, I intend to interpret and define options to functionally implement the individual criteria. Refer to the Orange Book for any detailed descriptions of the different criteria required at each security level.

DISCRETIONARY ACCESS CONTROL (DAC)

DAC enforces the "need-to-know" principle for an object or resource. At the C2 level, it can do so down to the granularity of a single user. Access permission to an object is assigned by authorized users only, usually the owner. Functional implementation is fairly easy. DAC is not, however, intended to keep information from prying eyes. It only protects against "need-to-know" violations and access is given by the "authorized" user, usually the owner.

a. Commercial Off The Shelf (COTS) products can satisfy some needs for DAC. The Air Force Assessed Products List (APL) [2] and the NSA Information Systems Security Product and Services Catalogue (ISSPSC) [8] contain several products and performance reports. They can implement DAC through software or hardware. Software is the easiest to use, but is also easily subverted in a PC environment. Some examples are ISAC 2200, Watchdog, and TRISPAN. Consult the APL and ISSPSC for the product that best suits your system.

b. Limit physical access to the system by locating it in a controlled space such as a vault or controlled access area. This does, however, create a dedicated system, minimizing the need for additional computer security.

OBJECT REUSE

This refers to the controlled reallocation of any storage space used by the computer, from floppy disks to optical drives. No information produced by a prior session is to be available to a subsequent user.

a. Most PCs and their peripherals use volatile RAM memory. This memory is erased by powering down the PC for 30 to 60 seconds [1].

b. Hard disks can use certain overwrite programs (consult the APL) to clear or purge data. Unless the PC resides in an open storage area, any hard drives with classified information should be removable.

c. Degaussers can effectively implement object reuse by purging or clearing magnetic media, but often render the media useless until reformatted. Consult NSA's Degausser Products List [8] for approved degaussers and AFR 205-16 for Air Force approved degaussing methods.

IDENTIFICATION AND AUTHENTICATION (I&A)

The Trusted Computing Base (TCB) will require users to identify themselves and then, via a password or some similar device, authenticate their identity. This is a well known requirement, but not the easiest to implement.

a. COTS software and hardware can provide some I&A, but they do have their limitations. The biggest glitch tends to be the ability to change the database the I&A module uses to verify passwords and their access limits with software schemes. Hardware schemes help to prevent this. Consult the APL and ISSPSC for details.

b. Biometric devices provide definite authentication but are very expensive for microcomputer applications. Also, if someone ever discovers a way to spoof them, your system will be wide open.

c. Portable one-time keys provide one more level of security by generating a new "password" for each access attempt. But they require the host and the user's authenticator to concurrently run the same algorithm and generate the same password. This is probably too costly for PCs.

d. If the system resides in a controlled area, the access requirements for that area may include manual I&A (i.e. a guard). This may satisfy your I&A requirements.

AUDIT

Audit trails are probably the best way to monitor system performance and user behavior. Automated audit trails are available for large mainframe systems, but hard to find for the PC environment. Several COTS packages attempt to implement audit trails, but the audit data can be modified through utility programs [2]. Some vendors are addressing the use of hardware implementations to avoid this problem.

a. Manual audit sheets cover some requirements, but rely heavily on the user to comply with security policy. Any user who does not record their actions simply goes unnoticed. Therefore, manual sheets may not work for your application.

b. COTS packages can implement audit functions but have shortcomings similar to the I&A modules. Auditing actions on the computer uses system resources. The more comprehensive the audit, the slower the system. PCs should not be affected by this problem, but the auditing of only certain events (pre-selection) uses fewer system resources. Consult the APL and ISSPSC.

SYSTEM ARCHITECTURE

The modification of TCB code or data structures will be restricted (TCB will be self-protecting) and resources controlled by the TCB (e.g. printers, etc.) will be labelled, just like other objects [7]. This requirement is for automated implementations. Most manual or functional features do not utilize TCBs, therefore not needing self-protection mechanisms. However, some automated features you have developed may have a functional implementation of self protection.

a. Control the master copies of the security programs if you are using automated features. Designate your security administrators and give only them full access to the security program functions (e.g. audit). If possible, limit the use of utility programs to security administrators, since these programs are the easiest way to subvert the security restrictions. Also, limit the personnel who are authorized to implement updates or improvements to the system.

b. Visually inspect the workstations on a regular basis. Ensure the peripherals connected to the system are the ones designated by the security program. Any configuration changes to the system should be approved beforehand.

SYSTEM INTEGRITY

Hardware and software features (i.e. diagnostic programs) shall be provided to periodically ensure the TCB is operating correctly. In other words, hardware diagnostic tests must be available.

a. If you are using TCB products, your security packages may provide programs to satisfy this requirement. Consult the vendor of your security program or the system documentation.

b. Use the standard operating system diagnostic programs provided by the vendor.

c. Utility packages like Norton Utilities provide several system diagnostics.

SECURITY TESTING

The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Functional testing is all that is required at C2.

a. COTS packages chosen from the APL and ISSPSC will have most of the required testing completed. Any variation from the suggested use may require additional testing. For example, the APL only assessed Norton Utilities 4.5 on standard Zenith systems. Use of the overwrite programs on a SUN workstation running DOS may warrant some additional testing to assure it works as desired. Also, APL products include some extra penetration testing beyond C2 requirements.

b. COTS packages not considered on the APL or ISSPSC will require testing. The vendor may provide help but use caution because this may bias the tests.

c. Any testing for functional or manual implementation schemes can be accomplished on-site. Ensure your procedures are correct and compliant with any appropriate regulations.

SECURITY FEATURES USER'S GUIDE

A description of the security features available to the user shall be consolidated in one summary, chapter, or manual.

a. COTS software will include a vendor manual. It will describe protection features, provide guidelines on use, and show how they interact.

b. If manual procedures are used, then the local Operating Instructions (OIs) can detail all necessary security steps.

TRUSTED FACILITY MANUAL

This is a manual for the AIS administrator and presents cautions about security functions and privileges they control when running a secure facility.

a. For TCB products the vendor can provide a manual for the security administrator. It should describe how to run the system securely and how to maintain the audit trail data.

b. Local OIs should cover any manual security procedures (e.g. who reviews the audit trails, who can update the system?).

TEST DOCUMENTATION

The system developer shall provide the test plan, procedures and results of the security mechanism's functional testing. Penetration testing is not required at the C2 level.

a. If COTS packages from the APL and ISSPSC are used, most testing requirements will be completed. Test results are available upon request.

b. Any features or functions developed on-site, including manual procedures, should have the test results from the required security testing documented and on-hand.

DESIGN DOCUMENTATION

Descriptions of the philosophy of protection and an explanation of its translation into the TCB will be available. In other words, whatever procedures you institute, document why you feel they are appropriate. In the case of TCB products, the vendor shall provide this information.

PUTTING IT ALL TOGETHER

Since every system and situation is so different, each answer to your security questions can and will be different and maybe even correct. Two completely different solutions may have equal effectiveness. Computer security in the PC environment leaves room for maneuverability and diversity. Do not rule out an option just because it is different.

Deciding what areas are lacking can be a chore itself. The Air Force is currently producing a series of guidelines and instructions for computer security to help field user's with this problem. Air Force Special Security Manual (AFSSM) 5011, Security Certification Guideline for Application Software [3], details minimum security requirements for application software. These requirements also apply to the system in general and can be used as a reference for the initial review of computer security in your system. AFSSM 5007, Functional Implementation of C2 Criteria, provides detailed guidance on all Air Force systems, including micros. Consult these and similar documents when they become available.

NCSC publishes the "Rainbow Series," a series of guidelines providing help to understand the Orange Book. They range from the Trusted Network

Interpretation (TNI), an interpretation of the Orange Book for networks, to the Password Management Guideline, a guide for use of password-based I&A systems.

When choosing your security implementations, take great care in noting how different options interact with each other. Using one package to accomplish object reuse and another to maintain audit trails may not be compatible. In fact, in the DOS environment, using one package to implement all the necessary security is almost always the best answer. In the near future, some companies will offer updates on their existing products to overcome the deficiencies noted in the APL and ISSPSC. Look for these. If you currently are using a product you like and feel it should be tested, submit it to the PERC (Air Force users only) or prompt the vendor to submit it to NCSC.

CONCLUSION

Computer Security is still a new field. No answer is perfect, and few answers are completely wrong. Use your judgement and the resources of AFCSC or your respective organization. There is always an element of risk. Our job is to determine what amount of risk is acceptable. Someday COMPUSEC will be as well defined as COMSEC or TEMPEST, but until then we must use ingenuity and common sense to ensure our microcomputers are secure.

Also note that while your system may run in dedicated mode and not require the C2 level of security, some basic precautions will help avoid trouble somewhere down the road. Address your security concerns and ensure your personnel are aware of them. Computer Security is the application of Risk Analysis and Risk Management. Simple education and awareness practices can help prevent data disclosure, protect the data integrity, and ensure the availability of service.

REFERENCES

[1] AFR 205-16, Computer Security Policy, 28 April 89.

[2] Air Force Assessed Product List, Jul-Sep 1989.

[3] Air Force Special Security Manual 5011, Security Certification Guideline for Application Software (Draft), 31 October 1989.

[4] Air Force Special Security Manual 5007, Functional Implementation of C2 Criteria (Draft), 24 January 1990.

[5] DOD Directive 5200.28, Security Requirements for Automated Information Systems (AISs), revised March 1988.

[6] DOD 5200.28-M, AIS Security Manual (Draft), 24 April 1989.

[7] DOD 5200.28-STD, DOD Trusted Computer System Evaluation Criteria, December 1985.

[8] Information Systems Security Products and Services Catalogue, Oct 1989.

[9] NCSC-TG-004, Glossary of Computer Security Terms, 21 October 1988.

Executive Summary

LIMITING ACCESS TO KNOWLEDGE AND INFORMATION

A Panel Discussion

Panel Chair
Robert J. Melford
R. J. Melford Associates
28195 Manchuca
Mission Viejo, CA 92692
(714) 830-1443
r.melford%compmail@intermail.isi.edu
r.melford@compmail.com

The Computing Ethics Subcommittee, of the IEEE Computer Society, has established a team of ethicists and technologists to encourage a continuing dialogue about these issues within the profession, and through out society, as a whole.

At this conference, we examine the exposures, costs, and benefits associated with systems designed to control access to knowledge and information. The relative extremes of [commercial] restricted access systems and [academic] open access systems are considered. The historical desire by society to provide a legal balance between these competing goals is reviewed. Finally, we explore the possible paths of influence between our ethical positions and future technologies.

Executive Summary

CONSIDERING THE IMPLICATIONS OF FUTURE TECHNOLOGIES

Dr. Ramon C. Barquin
Washington Consulting Group
11 DuPont Circle, NW
Suite 900
Washington, DC 20036

It's well established that new technical innovations, or technologies, often generate new ethical dilemmas, or old ones under new forms. As a result of this, IBM perceived the need to address a series of questions for which the answers were not readily apparent within the business community; questions dealing with human values and possible new ethical dimensions brought about by the advent of computer technology, and its massive adoption within society. These endeavors led to some joint undertakings between technologists and ethicists culminating in several seminars, two publications, and a convocation.

Some of the general results coming out of these efforts will be described, and insights will be brought out which might serve to understand better the implications of future technologies.

Executive Summary

CONSIDERING THE IMPLICATIONS OF FUTURE TECHNOLOGIES

Dr. Ramon C. Barquin
Washington Consulting Group
11 DuPont Circle, NW
Suite 900
Washington, DC 20036

It's well established that new technical innovations, or technologies, often generate new ethical dilemmas, or old ones under new forms. As a result of this, IBM perceived the need to address a series of questions for which the answers were not readily apparent within the business community; questions dealing with human values and possible new ethical dimensions brought about by the advent of computer technology, and its massive adoption within society. These endeavors led to some joint undertakings between technologists and ethicists culminating in several seminars, two publications, and a convocation.

Some of the general results coming out of these efforts will be described, and insights will be brought out which might serve to understand better the implications of future technologies.

Executive Summary

PATENT, TRADE SECRET, AND COPYRIGHT LAWS: ONE FACET OF THE GOLDEN RULE APPLIED TO LIMITS ON ACCESS TO KNOWLEDGE AND INFORMATION

J. Timothy Headley, Esq.
Baker & Botts
One Shell Plaza
910 Louisiana
Houston, TX 7002-4995
(713) 229-1561
t.headley%compmail@intermail.isi.edu
t.headley@compmail.com

1. In an ideal society, creators of new information would always freely share it, and, users of the new information would willingly, instantly, and eagerly pay a fair price for it.

   This assumes that the users would use the information to promote the health and well-being of others, rather than using the information to hurt or destroy

   Because we live in a less-than-ideal society, we need laws to (1) encourage proper payment for safe information, and (2) ensure proper protection of harmful information.

2. As knowledge increases, we should continue to build on our existing framework of laws in order to properly limit access to new information and to limit access to the organization of existing information. This discussion assumes the information is "safe".

   a. As early as 1432, Venice had laws limiting the use of new information.

   b. In 1787 the founders of our country put into the constitution a provision to limit the use of new information.

   c. The Supreme Court has consistently recognized that the use of information (intellectual property) must be limited.

   Limitations on the use of certain information necessarily implies the necessity for some limitations on access to that information.

3. Reminding ourselves of the ancient laws may help us in building the superstructure of our limitations on access to knowledge and information.

Individuals have made enormous changes in societies by actively pushing for new laws. We should continue to actively work to put high ethical standards into laws relating to accessing knowledge and information.

Executive Summary

SOCIETY RUNS ON TRUST

Ralph J. Preiss
IBM Mid-Hudson Laboratories
One Shell Plaza
910 Louisiana
Poughkeepsie, NY 12603
r.preiss%compmail@intermail.isi.edu
r.preiss@compmail.com

Whether we like or not, as computer professionals we are always faced with ethical problems. There are always limited resources which must be distributed equitably. Be it the departmental budget, if we manage a department, or the professional time we spend on our company's business, if we work professionally for a company. We can devote our time to our own self-interest alone, or we can devote it to our company's interest. Our colleagues, our managers, our company stockholders, all depend on us, trust us to uphold our professional pledge of doing a professional job. We, in turn, trust our company to reward us with job security, a pleasant and safe workplace, and a professional's pay.

In a larger sense, society expects that all professionals, from grocery clerks to bank tellers, policemen, physicians, bus drivers and sanitation men, etc. perform their duties within certain guidelines so that society can function smoothly. Chaos would ensue if the job each is expected to perform is not performed to expectations, or in a timely manner.

Because we provide tools for others, computer professionals are secondarily involved in many of the primary occupations being carried out to society's expectations. We provide algorithms for counting, measuring, auditing, deciding, and implementing decisions. Whether through programming or hardware design, we influence the outcome of other people's activities, and therefore we must do the utmost to ensure that accurate and correct decisions are employed, and that incorrect ones are flagged. It is the ethical thing to do. Society expects and trusts us to carry it out.

In the limited time available, I shall use some illustrations of ethics taken from computing applications. I shall explore ethical questions in decision-making regarding aspects of safety, analyze some ethical angles of intellectual property, stretch the definition of privacy to include the influence of computers, and touch on societal ethical norms for maintaining human life. In all cases I shall return to the need for professional ethical behavior and trust in the professional to make the system work. I shall try to share some insights on ethical questions as influenced by the advent of computers. Being a practicing engineer and not an ethicist or theologian, I shall quote individuals who are, along the way.

Executive Summary

OPEN ACCESS SYSTEMS: RISKS & RESPONSIBILITIES IN THE ACADEMIC ENVIRONMENT

Dr. Jane Robinett
Polytechnic University
Department of Humanities
333 Jay Street
Brooklyn, NY 11201
jrobinet%vm.poly.edu@isi.edu

This paper will deal with the need for and the protection of open access systems. Most frequently found in an academic environment, open access systems are vital to teaching, research and learning. Open access systems allow users to gain experience in a number of areas vital to their work and facilitate the easy exchange of information and experience which is necessary to research in many areas. They provide a rich working environment necessary to science, technology, industry, business and art. Universities would be hard-pressed to continue their work in education and research using more restricted systems.

But because of the large number of users of varying levels of expertise who have access to such systems, open access systems are increasingly vulnerable to accident and mischief and malicious intent. The question of how to protect such systems and who should assume the responsibility for system security is one that needs the careful, critical attention of all the members of the community.

Systems security can be approached from several directions: protective hardware and protective software are the two most familiar ones. But the responsibility for system security in open access systems should not be the unique burden of system managers, programmers and security experts. The user community shares responsibility for the integrity and security of the system which serves them, as well as other systems linked to that system. This is something which most users have little understanding of, and which even experienced professionals can understand or forget.

It is for this last reason that the public discussion of ethics as they apply to the world of computers has become so important. Unless users have a clear sense of their responsibilities toward the community of which they are a part, we cannot hope to protect open access systems from mischief, misuse and malicious damage.

Executive Summary

COMPUTER EMERGENCY RESPONSE TEAMS: LESSONS LEARNED

E. Eugene Schultz
University of California
Lawrence Livermore
National Laboratory
P.O. Box 808, L-619
Livermore, CA 94550

Richard D. Pethia
Software Engineering
Institute
Carnegie Mellon
University
Pittsburgh, PA 15213

Jerome R. Dalton
AT&T
Corporate Security
20 Independence Blvd.
Room 4B27
Warren, NJ 07060

Abstract

Since the 1988 Internet Worm teams have been formed to respond to computer security incidents such as network intrusions and computer virus outbreaks. This paper presents a description of three such teams, the Computer Incident Advisory Capability (CIAC), Computer Emergency Response Team Coordination Center (CERT/CC), and AT&T Corporate Security. The focus is on each team's structure, activities, and procedures. This paper also presents several high-level analyses of incidents and trends, as well as prescriptions for improved security.

The Computer Incident Advisory Capability (CIAC)

CIAC is the Department of Energy's (DOE's) Computer Incident Advisory Capability. It is a team of computer scientists at Lawrence Livermore National Laboratory given the task of assisting DOE sites faces with computer security incidents (e.g., hacker attacks, virus infections, worm attacks, and so on). CIAC is, in effect, an emergency response team for computer security problems within DOE, providing a centralized response capability. This capability is available on a 24-hour a-day basis to the DOE community (77 DOE and contractor sites and regional operations offices).

Activities

The primary activity of the CIAC team is to provide assistance to DOE sites when such assistance is requested. This assistance includes assessing the nature and extent of damage to computer systems, coordinating technical efforts to develop and collect software "patches" for problem resolution, advising site personnel how to perform damage control and recovery procedures, and providing direct assistance on complex technical computer security related problems. A closely related activity is coordination of efforts to respond to emergencies and to close computer vulnerabilities. This coordination activity involves DOE sites, other Government agencies, computer vendors, and other similar response teams. A third activity is to alert the DOE community about computer security threats; particularly network attacks, computer vulnerabilities, and new computer viruses. In addition, CIAC has developed and presented a computer security incident response workshop that teaches others how to respond to computer security emergencies.

Team Composition

The CIAC team needs to be able to handle a wide range of incidents. Most of these incidents involve Internet and largescale DECnet attacks, as well as virus infections in small systems, giving us a rather large domain to cover. We currently have expertise in UNIX, VMS/DECnet, VM, DOS and Macintosh operating systems. Because we are budgeted for five staff, however, we are unable to have much redundancy in any one area of expertise. In some cases, we need to refer those who contact us to someone up the "guru chain," or turn to another incident response team for advice or confirmation of a solution.

Constituency

Nominally, our constituency consists of the DOE sites and regional operations offices. In actuality, however, the boundary lines for our constituency have not been clearly demarcated. Most of the middle- and high-level management at DOE sites and virtually everyone at DOE operations offices will turn to CIAC for assistance (or at least will inform CIAC of major events) if they learn of a noteworthy computer security incident. At the end user and system manager level, however, this is not necessarily the case. End users and system managers at DOE sites faced with specialized problems may contact some other response team first, if not exclusively. This situation is by no means a catastrophe, and, in fact, we have what we believe is a large and very loyal constituency within DOE. The fuzzy boundaries of this constituency has, however, created some complications. We often do not learn of the site's problems rapidly enough, if at all, which impedes our ability to coordinate incidents across DOE.

A somewhat different concern is that because CIAC bulletins are rather widely distributed outside of DOE, we discover that many from other Government agencies and commercial industry turn to us for help. Although we are glad to assist, these numerous requests from outside the DOE community create a good deal of extra work for us. We suspect that the fact that we are not currently identified with any specific network is a good part of the reason that our constituency boundaries are somewhat unclear. Our emerging relationship with the Energy Sciences Network (ESnet) may help CIAC to establish clearer constituency boundaries. We also suspect that there are some people within the DOE community not familiar with our charter who incorrectly fear that we exist to report people with poor computer security practices to DOE Headquarters--accordingly, some of these people do not turn to us for assistance.

Activities

During the first few months of this year one of our DOE Headquarters sponsors analyzed CIAC's activity in detail. One of the findings was that, although CIAC deals with a wide range of incidents, since its inception CIAC has dealt with virus problems more than any other single type of incident. Many of these incidents are technically demanding, and require CIAC team members to disassemble virus code, work closely with vendors of virus detection and eradication software, and to quickly advise the DOE community of imminent virus (and trojan horse) threats. A close second in frequency are network-based attacks. When the WANK/OILZ worms attacked the High Energy Physics Network (HEPnet) and the ESnet in October, 1989, for example, CIAC developed effective immunization scripts and coordinated responding to this threat across DOE sites and other agencies and organizations. CIAC, in coordination with CERT, has helped numerous DOE sites with Internet hacking attacks since CIAC's inception. A considerably smaller portion of CIAC's activity has been devoted to activities such as investigation insider attacks.

CIAC's efforts have not, however, been exclusively reactive. CIAC team members have devoted considerable effort for developing a two-day workshop on dealing with incidents. We reasoned that there will, at best, be fewer than one dozen people on CIAC, so the technical resources we provide will necessarily be limited. Within the DOE community, however, are many very technically capable individuals who, with a little additional training, could do many of the things of which CIAC team members are capable. We wanted at the same time to raise the awareness level within the DOE community of the types of problems that lead to computer security incidents, as well as appropriate preventative measures. The CIAC workshop has been an overwhelming success both within and outside of DOE, and has greatly expanded our ability to serve our constituency. In a parallel vein, we have developed a now rather complete set of incident handling procedures to be distributed throughout the DOE community. We are developing tools, such as a tool to identify unpatched vulnerabilities in an operating system. Finally, we spend a considerable amount of time dealing with vendors to identify and close vulnerabilities, and to keep communication channels open.

What Have We Learned?

Many of the people whom we help are both very knowledgeable about computer security and are very competent practitioners of this trade. A frustrating "fact-of-life" for CIAC team members, however, is that many incidents in which CIAC has assisted could and should have been prevented. There is still a widespread naiveté about sound security practices among users of small, standalone computers. Users of small systems, for example, tend to be careless about exchanging removable media or in using electronic bulletin boards. Too often users who know or strongly suspect their machine has a virus will fail to take appropriate action, resulting in widespread outbreaks of that virus. Other users arm themselves with arsenals of virus detection and eradication software, overlooking the fact that no such software is foolproof and that following sound procedures for preventing and dealing with virus infections are just as important as any software tools.

Users of larger, networked systems may know more about sound security practices, but they too often fail to put their knowledge into practice. For example, these users frequently choose weak (very guessable) passwords, perhaps in part because so many system managers and administrative managers have no password policy. Many if not most system managers do not learn about system vulnerabilities and fix them. Complicating matters is that people seldom plan adequately for responding to an incident, which generally makes this process slower and more complex. Too few organizations have a complete, written contingency response plan accompanied by regular mock incident handling exercises, and too frequently the competent technical personnel are unable to be effective due to ambiguity of procedures and/or scarcity of technical resources available to them. In the same vein, we find a pervasiveness of what we call the "Orangebook tunnel vision mentality"-- "I'll set my system up so that I can pass inspection" at the expense of planning to deal with practical, day-to-day computer security problems. Perhaps becoming more aware of the *real* threat out there--teams of highly organized intruders who are often more aware both of users' tendencies to use weak passwords and of vulnerabilities than we are!

The Computer Emergency Response Team (CERT)

Shortly after the Internet worm of November 1988, the Defense Advanced Research Projects Agency (DARPA) created a Computer Emergency Response Team (CERT), with a Coordination Center (CERT/CC) at Carnegie Mellon University's Software Engineering Institute (SEI). The CERT/CC at the SEI supports two different communities: Internet users, and developers of technology that is available on the network, such as Unix and networking software.

Since its inception, the CERT/CC has responded to a continuous stream of reported security incidents. These include reports of intrusions and worms as well as reports of vulnerabilities and suggested fixes for problems. In dealing with these reports, the CERT/CC has learned effective methods of operation as well as trends in security incidents.

Methods of Operation

To deal with computer security incidents across a broad community, a response team is most effective it if has certain basic characteristics.

●Skills Mix: Response team members must have the technical expertise required to handle the majority of day to day incidents as well as working relationships with trusted technical experts in the user and vendor communities. In addition, since the resolution of a particular incident often requires more communication and coordination than technical work, team members should have excellent communications skills and have the ability to interact effectively with various types of people: technical staff, management, investigators, government officials. Each team should be supported by a strong administrator who can assist in organizing and tracking countless details as well as producing detailed reports.

●Service Orientation: Since a response team typically assists in the resolution of problems in systems that the team itself does not own or operate, it is critical that the team adopt a service, rather than an enforcement, orientation. Authority is vested on the team by its constituents and the vesting will only occur if the team develops and communicates its services and consistently delivers those services.

-Critical Mass: A response team must be able to conduct both reactive and proactive activities and the team must be adequately staffed to handle both. In addition, prolonged periods of reactive activity can easily lead to "burn out" and each individual should be able to cycle through periods of reactive and proactive work.

●Visibility: Computing communities continually grow and change and a response team must make extra effort to insure it continues to be visible within its community. Utilizing low cost but high leverage mechanisms (e.g. presentations at established conferences, electronic newsletters with wide distribution) to publicize the activities of the team will help insure problems are reported to the team when they occur.

●Availability: It is critical that the team be available to its constituents on a 24 hour-per-day basis. At a minimum, there should be a well publicized hotline as well as an electronic mailbox to capture information at any time.

●Contacts: An individual response team typically cannot resolve an incident without assistance or coordination with outside individuals or organizations. A contact database should be established that includes other CERTs, technical experts, site administrators, investigators, network operation centers, etc. Significant effort must be expended to insure the contact information remains valid over time.

•Facilities: During crisis situations, team members require information at their fingertips and the ability to rapidly communicate with their teammates as well as with others outside the team. Each team member should be equipped with a workstation that allows rapid access to information repositories as well as a phone desk set with multiple lines (including the teams hot line.) In addition, an off-line system should be established to archive and manage sensitive information. Finally, the team will be most effective if team members are physically co-located to enable quick conferences and one-on-one communication.

Incident Trends

In dealing with incidents on the Internet, certain trends have become apparent.

•System Administration: Weak account management, password management, and configuration management practices account for the majority of the reported incidents. Many sites have yet to adopt practices that insure users protect their accounts and use passwords that are difficult to guess. Failure to change vendor supplied default passwords, or to password protect accounts that are shipped without passwords, is an especially persistent problem. In addition, configuration management practices that delay installation of new versions of software that contain corrections to computer security problems leave many sites vulnerable.

•Vigilance: Over 60% of the sites that suffer intrusions learn about the problem only when other sites or the CERT/CC notify them. The contents of files left behind by intruders on a particular system often point to systems at other sites that have also been compromised. Intruders often target accounts that are infrequently used and attempt to disable system logging mechanisms to disguise their use of the compromised system.

•Vulnerabilities are Shared: As system intruders successfully gain access to systems which have weak passwords or systems where known vulnerabilities have not been closed, they often share information on vulnerabilities in these systems with others. Likewise, as intruders discover new vulnerabilities, information is quickly exchanged through various bulletin boards and other electronic forums. As a result, many communities of users quickly become vulnerable. Traditional methods of dealing with vulnerability information, including closely protecting information on the existence of the vulnerability, are not effective once intruders have learned of system weaknesses.

•Boundaries are Ignored: Unauthorized activity is not typically confined within traditional boundaries (e.g. national, network, computer architecture) and intruders frequently cross these boundaries several times per incident. While many incidents occur because of weak practices or design or implementation deficiencies, resolution of the incident requires more than a technical solution. Communication of threat and vulnerability information across computing communities is essential to resolving specific incidents and improving the security of operational systems.

•Things are Planted: Intruders often plant backdoors and trojan horses in the compromised systems. Back doors are used to gain access to a system in case the original vulnerability is closed. Trojan horses are most often planted to capture additional account and password information: information that may often be useful to the intruder long after the original intrusion and also allow the intruder to gain access to systems where users have additional accounts.

AT&T Corporate Security

Background

Prior to 1987, AT&T experienced a few isolated attacks on their computer system and networks. However, during the early part of 1987, attacks from external computer hackers increased dramatically and AT&T was quickly thrust into the Emergency Response Process. The need for an organized approach to this problem was apparent, and efforts to develop, document, and test a plan were initiated.

The urgency to facilitate this project became more urgent in November, 1988 when the "Morris (Internet) Worm" attacked approximately 6000 computers on the Internet. Although AT&T computer systems were not victimized, the emergency response plan was exercised to the extent that emergency contacts and subject matter experts were called upon to quickly determine the nature and extent of the threat, identify potentially vulnerable processors, develop effective countermeasures, and ensure that all computers were analyzed. Executive status reports were written and Public Relations contacts were alerted.

Since the Internet Incident, activities associated with the development of a Crisis Management Plan have accelerated. Although the process of emergency planning is never complete, significant strides have been made toward improving AT&T's ability to quickly respond to computer and network emergencies. This experience has also been very educational.

Building a Constituency

During the course of dealing with several computer and network incidents over the past few years, AT&T quickly learned the need for establishing and maintaining constituent relationships, both internal and external to the Corporation. The profile of the constituency may differ somewhat from one emergency to the next, but the success of any emergency response team is clearly dependent on its ability to contact key personnel and subject matter experts in a time of crisis.

Preparation

The process of building and maintaining constituent relationships is ongoing. Periodic communications with emergency contacts and subject matter experts must be done to ensure a quick and effective response to any potential problem.

Test the Plan

Although a full emergency cannot be tested as it would under live conditions, simulated scenarios should be conducted to identify weaknesses in the plan.

Quick Response Essential

The success of any emergency response team depends on its ability to quickly identify, isolate, eliminate, and further prevent any computer or network related attacks.

Documentation

During a given emergency, many people gather important information about events surrounding the incident. Every member of the team should chronologically record their activities. This information provides important details that not only determine a course of action, but provide a documented history of the event.

Post Emergency Analysis

A great deal can be learned during the post emergency analysis meeting. Weakness in the process become apparent and the review process can sometimes yield new and creative solutions to old problems.

Summary

Discussions about lessons learned are important because they give us an opportunity to share our experiences with others who have probably encountered the same or additional problems.

Overall Summary

Each of three managers of incident response teams agree that the task of responding to computer security incidents is a challenging (if not overwhelming) job. It is important to provide a quick and effective response to incidents, and to develop and follow an effective set of plans or procedures. It is also critical for each team to establish cooperative relationships with a constituency, and to maintain a service orientation in dealing with that constituency. Dealing with numerous incidents (especially Internet attacks) has shown that there is a need for greater awareness of and adherence to sound security practices.

DISCERNING AN ETHOS FOR THE INFOSEC COMMUNITY: WHAT OUGHT WE DO?

by

Eric V. Leighninger
Senior Software Scientist

Dynamics Research Corporation
60 Frontage Road
Andover, Massachusetts 01810
Phone: (508)475-9090
Arpanet: eleighninger@drcvax.af.mil

Abstract: The development of an ethos for the INFOSEC community is a necessary activity if we are to create an adequate and proper framework for dealing with the ethical issues of personal and institutional behavior relating to information security matters. The objective of this paper is to identify open ethical issues and potential modes of analysis that might aid the INFOSEC community in creating the building blocks of an ethos and a framework for ethical discourse. The paper also identifies problems and issues deserving serious critical analysis in the process of discerning the appropriate ethical model for the INFOSEC community.

Topical Area of Paper: Ethics and Issues

- Computer Use and Abuse
- Relationship of Ethics to Technology - Standards of Ethics in Information Technology

Key Words: Ethics, ethos, norms, behavior, technology, information security

1.0 The Need for an Ethos and a Framework for Ethical Debate

The responsible use of computer systems, and the associated computer security technology being developed to protect them, has raised concerns involving individual ethics and trust, institutional uses of privileged data, and how protection mechanisms themselves might be abused or misused. The idiom "between Scylla and Charybdis" seems to aptly describe the tenuous balance between providing for computer-based security of sensitive data and information, and safeguarding individual and institutional rights to privacy and correct use of privileged information. Balancing concerns such as privacy protection, information security, and the positive and negative implications of security technology developments has created a pressing need for structuring debate and engaging in meaningful moral action in these areas. Denning et al in [1] stated this need succinctly in saying "Our goal must be to create an atmosphere that motivates people to behave responsibly and with confidence that their rights and information assets are protected."

This goal statement raises many questions; for example:

What should this atmosphere or context of action consist of?
What factors will motivate people and institutions to behave responsibly?
What criteria or norms should be applied to assess or even recognize responsible behavior?
What values or goals are worthy of evaluation as they pertain to security technology development and use?

The pursuit of a goal such as Denning's, and the discernment of means to obtain it, is one of normative ethics. It is the purpose of this paper to identify significant open ethical issues, to suggest methods of analysis that the computing community should be considering, and to offer a proposed framework for engaging in such a discourse and analysis to aid in the formation of an INFOSEC community ethos. It is the author's opinion that the specific tools, language, and modes of reasoning used by social ethicists can help to frame the problems posed by security technology in a manner that will help the computing community reconcile the conflicting set of "oughts", both personal and institutional, which are arising as issues today. This reconciliation will possibly create the foundation for an ethos for the INFOSEC community in which conscientious individuals and responsible, accountable groups interreact in right, good, and fitting ways.

2.0 Forming the Basis for Ethos: Norms and a Framework for Responsible Ethical Discourse - How Should We Respond?

The type of questions arising in the use and evaluation of security technology, as noted earlier, generally compare to one or more of the three types posed in ethical reflection:

What is the right thing that ought be done?(The deontological view)
What is that course of action which will be good in some sense? (The teleological or utilitarian view)
What is the fitting or appropriate action in a particular case? (The contextualist view).

Ethical theory offers a wealth of techniques for formulating arguments and establishing premises for dialogue within the boundaries of any or all of the above categories. Since ethical reflection involves the provision of a prescriptive approach to behavior, the approach the security community takes to such questions must be directed towards developing a strategy for defining the operating norms of the information security community and how the members behave according to them. The author would like to recommend two principal lines of research, which can serve as a starting point in helping to provide a framework for ethical debate in the security community. Suggested open problems in these two arenas are presented in the next section. The lines of work proposed for consideration in this paper are as follows.

1. Develop a set of canonical ethical problem statements relating to INFOSEC that are derived from experience for analysis and evaluation. The case studies provided in evaluation of the INTERNET worm and the penetration of the Lawrence Berkeley computers (references [2] and [3] respectively) are representative starting points. Parker et al [4] provide another useful source of potential paradigms in their case studies and scenarios on ethical conflicts drawn from the technology and business

arenas. These paradigms serve to highlight key issues, provide a basis for developing tools and techniques for problem resolution, and help structure the philosophical debate by focusing attention on specific problem types. This set of canonical problems can be used over time to develop a taxonomy of ethical questions pertaining to information security.

2. Examine critically the major streams of ethical thought in the teleological, deontological, and contextualist schools (see, for example, [5]) for questions and approaches addressing issues similar in content to those considered in the security and privacy contexts. Both classical and contemporary ethical works have lines of logic that may be relevant. Just as philosophic concepts such as epistemology and logic have touched on modeling in security and artificial intelligence, so possibly, the works of the British school in semantical analysis may prove of import to analysis of such terms as trust and privacy and their linguistic use in the security literature. Possibly as well, the Austrian axiological (value analysis) approach may be germane to this discussion of concepts such as "expectation of privacy" as a value-based norm for behavior. Also security practitioners should draw on contemporary work in the areas of bioethics and nuclear policy, where appropriate; particularly with regard to questions of the means and ends of technology development. Interaction with the legal philosophic community, particularly in regard to questions regarding rights and how they are defined, shows evidence of possible benefits.

This initial work will provide practitioners with the typology of problems to be addressed at this time, the tools for rational discourse, and the initial building blocks of an ethos. These initial building blocks provide a solid philosophical base upon which to derive a useful, systematic, goal-oriented methodology for identifying norms for ethical behavior. Such a methodology is a starting point for establishing concrete guidelines for instituting behavioral norms and resolving ethical conflicts. Some immediate areas of application of such research would be the generation of individual and organizational codes of conduct pertaining to INFOSEC concerns; development of broad-based, general standards of ethics regarding information protection to support law and policy formation; and building curricula for ethics courses for computer science students. Some initial work in these areas has already begun (see references [6], [7], [8]). More remains to be done, however, before an identifiable ethos for the INFOSEC community is realized.

3.0 Current Problems Deserving of Consideration

Listed below are a number of ethics problems the examination of which should shed some light on how the INFOSEC community may need to proceed in establishing the norms and behavior guidelines for individuals and organizations.

3.1 The Relationship of Individual Ethics and Organizational Ethics

The discussion regarding ethical behavior in the security community has so far principally focused on that of the individual's personal code of moral conduct and integrity. Little discussion if any has occurred to address the concept of reciprocity of institutions *vis a vis* individuals and the institution's obligations in areas such as policy formulation, technology employment, and their potential adverse sociological impacts on individuals as members of those institutions. These concerns can be examined by viewing personal ethics and institutional ethics as opposite ends on a spectrum of personal to collective ethical relationships. A sample illustrating such a spectrum of ethical problems ranging from primarily personal in nature to primarily collective or institutional in context is given below:

1. The hacker problem profiles the INFOSEC problem as primarily one of personal ethics *vis a vis* the institution.

2. The privacy invasion problem posed by Campbell in [9] profiles the question of personal ethics in the use of sensitive information resident in a set of government data bases. In this case the information is given by a citizen under an expectation of confidentiality and responsible care in its use, not for the personal, unauthorized purposes of a government employee with access to that information.

3. The user or employee dislocation phenomenon (employment of security technology itself produces a situation prone to security abuse due to the negative performance or sociological impact of the technology on the users of the system being protected.), postulated by Denning in [1], illustrates the dual responsibilities of the individual and institution regarding ethical behavior stemming from the dialectic created by the use of surveillance and other threat-monitoring procedures to limit computer system misuse and abuse.

4. The failure of institutional repositories such as credit bureaus and insurance companies to properly protect and disclose sensitive medical, insurance, and financial information is a question of institutional ethical responsibility *vis a vis* the individual.

5. The unauthorized use of privileged information given to the government by its citizens, such as tax data, is an example of institutional abuse of trust, as is surreptitious computer monitoring of individuals through unauthorized computer-based intercepts and wiretaps and internetted data banks.

After considering the above model, a question that comes to mind is: Are there ethical standards that may be imposed on individual behavior in the areas of protection and dissemination of information in computing systems, but from which institutional entities such as governments or companies are exempt? For example, in what ways does the use of computers on the part of an individual to draw inferences regarding government data differ from the use of computer-based data on the part of the government to infer private information about an individual without his or her permission? The use of a spectral view of concerns may aid in discerning the "shades of gray" dimensions of a problem and the proper balance of right, good, and fitting in our responses. Such analysis will also promote the development of a professional code of ethics for security professionals and organizations involved in computer security to provide guidance in the resolution of this and related questions. That such guidance is much needed has been recently reiterated in a related context by Shore in [10] in his discussion on certification of software professionals and design codes for "trustworthy" software.

3.2 Security Technology: Its Development, Use, and Abuse

Computer security technology development has created a Damoclean situation regarding its use, as exemplified by the concerns raised in [1] regarding the implications of surveillance and threat monitoring techniques on privacy rights, and the issues raised by Neumann in [11] regarding the technology and social gaps between computing mechanisms, policies, and social behavior of user communities. The issue of software warfare and the use of virus technology for offensive military ends raised in [12] poignantly illustrates some of the potential ethical dilemmas the military software community may be facing in the near future. The challenge in the technology arena is to develop an appropriate model to deal with the enablement and exploitation dimensions of technology used in the security domain.

3.3 Semantical Issues Arising in Security Parlance and Their Impacts on Normative Ethical Constructs

Semantical analysis of terms such as trust, risk, and security as they apply to ethical norms is needed. For example, is privacy a value or a right, and why? What these terms mean and what they don't, as well as questions of shades of gray distinctions (e.g., the Office of Strategic Services (OSS) concept of trust used in WWII -- that is, some intelligence sources were more trusted than others to truthfully convey reliable information) should be articulated. The concept of implicit trust in human relationships and explicit trust criteria for computing systems should be compared and contrasted. Bok, in her definitive works on lying and secrecy ([13] and [14]), offers analyses of these concepts which deserve consideration. As noted earlier, such considerations will aid in developing a taxonomy of ethical problems related to information security. An analysis of how emerging norms can be discerned and developed would shed light on how to deal with concepts such as "expectation of privacy" as either an absolute or relative value-based norm, or whether productivity as considered in [1] is an appropriate utilitarian measure of corporate choice in the use of security measures in a computer environment.

Privacy and security are, in a sense, opposite sides of the same coin. Development of an effective conceptual framework to address both may be accomplishable by using the concept of an ethos wherein both individuals and institutions are viewed in an ethical context as equivalent. Mutuality, trust, integrity, and responsibility could be addressed in this setting. This would require development of a concept of "person" status for both individuals and institutions in manner similar to that used by legal community to deal with the corporation concept.

4.0 Conclusion/Summary

It is clear there is much to do in the attempt to build an ethos germane to the needs of the INFOSEC community. The field of normative ethics has much to offer in helping us accomplish our task. To date, most debate in this area has been in the area of privacy rights and the identification of some of the more salient problems pertaining to the unauthorized access and use of systems. However, much remains to be done in terms of establishing norms and reasoning paradigms for the complex issues of today's information and security processes. The research agenda and problems laid forth in this paper, with no claim to sufficiency to the total task at hand, are ones which the author believes will help structure debate on this highly important topic and analysis of which will form the basis of a systematic and philosophically sound methodology for identifying pertinent INFOSEC ethical goals, and norms. Once identified, these goals and norms can then be manifested in specific, practical guidelines and standards for individuals and institutions in the areas of information security.

Acknowledgements

Thanks go to Peter Denning and Marlene Campbell for suggesting this essay, Mary K. Dahl for a stimulating discussion that helped crystallize thoughts on some of these issues, and to Dorothy Denning and Peter Neumann whose articles provided a good initial foundation for reflection in this area. A special note of thanks to my colleague, Lucy Sprague, whose sensitivity to ideas and keen editorial skill made this a better paper.

5.0 References

[1] D. Denning et al, "Social Aspects of Computer Security", Proceedings of 10th National Computer Security Conference, September 1987.

[2] J. Rochlis and M. Eichin, " With Microscope and Tweezers: The Worm From MIT's Perspective", CACM, Vol. 32, No. 6, June 1989.

[3] C. Stoll, "Stalking the Wily Hacker", CACM, Vol. 31, No. 5, May 1988.

[4] D. Parker, S. Swope, and B. Baker, Ethical Conflicts. Wellesley, MA: QED Information Sciences, 1989.

[5] V. Bourke, History of Ethics. New York: Doubleday and Company, 1968.

[6] G. Watt (Major), "Malicious Code: An Ethical Dilemma", Proceedings of the 12th National Computer Security Conference, October, 1989, pp. 542-552.

[7] W. Maconachy, "Computer Science Education, Training, and Awareness: Turning a Philosophical Orientation into Practical Reality", Proceedings of the 12th National Computer Security Conference, October 1989, pp. 557A-557I.

[8] H. DeMaio, "Information Ethics, a Practical Approach", Proceedings of the 12th National Computer Security Conference, October 1989, pp. 630-633.

[9] M. Campbell, "Security and Privacy: Issues of Personal Ethics", Proceedings of the 10th National Computer Security Conference, September 1987.

[10] J. Shore, "Why I Never Met a Programmer I Could Trust", Communications of the ACM, Vol. 31, No. 4, April 1988.

[11] P. Neumann, "Privacy vs. Security", Panel on Individual Privacy - Technical, Social, and Legal Issues, IEEE Symposium on Security and Privacy, 18-20 April, 1988.

[12] S. Boorman, and P. Levitt, "Software Warfare and Algorithm Sabotage", SIGNAL, May 1988.

[13] S. Bok, Lying: Moral Choice in Public and Private Life. New York: Vintage Books, 1979

[14] S. Bok S., Secrets: On the Ethics of Concealment and Revelation. New York: Vintage Books, 1983.

VIRUS ETHICS: CONCERNS AND RESPONSIBILITIES OF INDIVIDUALS AND INSTITUTIONS

John R Cordani, EdD.
Asst Prof, Management Science
Adelphi University
Garden City, NY 11530

Douglas Brown, OHC
Holy Cross Monastery
West Park, NY 12449

Abstract: This paper is a discussion of the ethics involved in the study and propagation of self-replicating computer code. Several conclusions are reached which indicate a shared responsibility for viruses between the institution and the individual virus author. Suggestions are offered particularly to institutions of higher learning for appropriate actions which should provide for responsible exercise of academic freedom as well as individual and institutional responsibility.

Ethical Framework

The selection of an ethical reference point leads unavoidably to social and economic consequences of great import. Information system managers and users, society-at-large, must come to terms with at least two critical ethical issues: Why does information technology exist and may I put the interests of all others in the community at risk in pursuit of my individual goals? "...how far may we knowingly go in violating, or just putting at risk, the interests of others in our projects- to determine the casuistics of responsibility and can in general not be laid down a priori in the doctrine of principles..."[7 p.35].

It would be well to place these questions in the context of other intellectual activities. One way of describing the human enterprise is as the process of bringing order out of chaos. Our survival, well-being and fulfillment, in short, our common good follows from the establishment and maintenance of order. Our creative efforts are ordered properly only if they tend to the common good. We imagine, we plan, and we execute. Our works, we believe, allow us to exercise dominion over the physical world and to expand the world of the intellect. We seek to bring order and purpose into all that is before is; our tools, out technology, exist to enable is to further order the world for man's purposes. Technology, therefore exists to serve the common good. The freedom to explore and expand the boundaries entails the responsibility of respecting the structure in which the freedom is exercised.

Information technology exists through the creative efforts of individuals operating in alone or in concert to resolve problems and enhance the ordering of the lives of individuals, organizations, even societies. Information technology is the result of the creative efforts of the individual mind, the result of intellect applied to problem resolution of the highest order.

These characteristics are most evident in the examination of coding segments, where indeed, individual thought processes make 'program' a unique creation of the individual mind. The importance of the technology lies in its ability to promote the solution of problems for the individual and the society. It is this driving force, the shared use of the technology to serve a common good that creates a dilemma in developing a posture in reference to computer viruses.

The resolution of this dilemma must be built upon the foundations of the technology's reason for existence. Information technology, like no other technology before, has an ability to liberate man from lower level labors of both a physical and an intellectual plane, provide humans with new opportunities for choice and release the mind from the monotonous world.[1 pp 61-62]

If information technology is a 'common resource', what rights does the creator, the explorer, the student have to place that resource at risk? The seriousness of this question can perhaps be grasped with the recognition that "...Terrorists are trained to do damage to computer systems in a short time.[3 p 83].

Social Impact of Technology

Within Western societies a pervasive need for information systems has become the rule. Critical resources are now possible to control only through the use of information, accessed via computer technology, to provide for the common good. Resources as basic as water and food are controlled in their preparation, their inspection, and their distribution largely through the operation of computerized information systems. Basic human services such as fire, hospital, and police services are controlled by distribution systems reliant on computer information systems. Banks [11], insurance companies and the Social Security Administration are totally dependent on computer information systems to function. We all are dependent on the survival of these systems to maintain, in an absolute sense, our individual and collective lives. In short, these information systems are no longer just labor saving tools, but have become part of the infrastructure of the society. Thus, that which strikes at the information system strikes at the common good. With computer viruses, the technological trick becomes human tragedy.

The Nature of the Trick

Computer viruses share many of the characteristics of life forms; indeed viruses , given a reproductive cycle may be viewed as computerized life-forms. Our current difficulties with viruses in effect are the result of their ability to replicate and migrate throughout systems. Murray [9] makes a strong argument that "...the behavior of computer viruses is similar to that of an infectious process..." At the present time a crux of the problem is that, unlike most life forms, host programs and host systems, do not have the ability to produce anti-bodies for themselves to protect themselves from hostile life forms. When a virus is introduced into a system it is introduced into a 'friendly' media where it grows and replicates without restraint. What is the responsibility of the creator of the infection?

Consider an analogous field to work: biogenetics. In biogenetics life forms are manipulated, instruction sets are modified, in fact self-replicating organisms are created as the object of the research and production effort. The greatest control possible is placed on experimentation both internally to the actual experiment, by the experimenters, and externally by society. There is universal recognition that uncontrolled release of new life forms into the ecosystem is fraught with uncertainty consequences. The possibility of the creation of cataclysmic event chains is understood. The root of the destructive nature of the event chains lies in the inability of the ecosystem to respond to new organisms injected into the system without the system developing control mechanisms to contain the new organism's reproductive cycle. In biogenetics experimentation is conducted in providing for the 'worst case' scenario. Researchers, their laboratories, and their sponsoring organizations are held accountable to the state and society for their control of material. There is a clear recognition of individual and corporate responsibility grounded in a notion of responsible science. Should not programmers be so responsible?

Social Impact of Computer Virus Attacks

Campbell, [3], Kask [8], Gibson [6], and others suggest that the greatest consequences of the virus program proliferation may lie outside the direct damage wrought by a virus but in the creation of a lack of confidence in the process of computerizing. To date system design has concerned itself with the production of systems which were increasingly 'friendly' and open to the user. We may now be seeing a movement, almost unique in this area, to close systems and prevent the free and open exchange of program and data, within and without systems. How much of our energies and machine cycles are going to be directed to virus detection and prevention? What will we all loose when cycles are slower to detect viruses? What will we loose in the loss of programming staff, diverted now to prevention rather than useful problem resolution? The price will be paid in a constant drain of human and machine resources from more productive efforts.

These diversions of computational resources are inevitable given our awareness of the consequences. Our exposure to virus risk results in: loss of confidence, increased absolute risk, and increases economic risks. Degradation of confidence in information systems results in the system user discomfort and hesitance as a consequence of increased anticipation of system failure. Users may begin to exhibit unrealistic apprehension and expend resources 'recovering' from non-existent infections. Given the absolute dependence many users have on computerized information systems, it is understandable that their behavior may exhibit strains of paranoia. Users face an increasing risk that a virus might be present on their system and will corrupt and halt information processing. The presence of a virus on these systems will shortly, if not already, result in the loss of human life, and perhaps on a far more widespread scale than we should like to imagine.

Although critical systems are now susceptible to virus infections and virus caused failures, forecasting is still difficult. Already the realization of an absolute virus risk is causing and, indeed, has caused systems designed to speed the flow of academic information, slowed. Users are becoming restricted in the free and open exchange of ideas and research as networks must react to the threat of viruses [5]. The economic impact of computer viruses can be measured in the diversion of computer machine cycles and programmer skill into the identification, location, and neutralization of virus infections within information systems.

Contributing Causes

There exists a technological imperative in the western world which has driven science and technology on an endless quest to accomplish the execution of that which is possible and to create the real possibility to create that which is imaginable. We are the animal of tool use. We imagine, we plan, and we execute. Our works, we believe, allow us to further conquer the world, the physical world and the world of the intellect. We seek to bring human order and purpose into all that is before us. Human enterprise as it exists in the image of God; a constant striving toward order from chaos the aim of human enterprise is to restore order - the precepts of free enterprise [liaise faire] were conceived as releasing the individual to strive towards order and the common good. Our tools, our technology, exists to enable us to further order the world for man's purposes. We, as a species, are driven to conquer the world in order to satisfy our need to service as individuals and as groups. The development of a computer virus is at its root counter to the common good it is a deliberate and calculated move towards the creation of chaos.

Randomness

The chaos created by the introduction of computer viruses into information systems results in user ability to function. The system user has reacted to viruses much as society has reacted to other random acts of chaos, acts such the poisoning of Tynanol provide insight into the consequences of virus threats. As in the Tynanol incidents, the user community comes to distrust the entire system. This distrust contributes to false reports of virus infections. The false reports, the perception of virus infection, cause the information system user and support staff to expend time and energies in correcting non-existent problems, looking for causes which are often truly counter-indicated. The threat of a virus, its random nature and elusive character can disrupt operations even when no infection is, in fact, present. Users are becoming quick to ascribe system failures to viruses and exhibit fearful behaviors, rather than seek more mundane explanations and remedies for their operational difficulties.

Perpetrator Defenses

The virus creator appeals to ignorance of the effect and seeks to have his case considered on intent rather than on probable or actual effect. The case, for the perpetrator, is framed in 'particular ignorance.' The excuse of particular ignorance is not ethically, morally, or legally binding. As in the analogous biogenetic area, the experimenter is accountable for the actual results of the experiment.

A strong case can be made for the conducting of computer virus research in a 'model' environment. Mathematics and the sciences have historically conducted research through models, models that might first be expressed logically, mathematically as axioms and theorems of formal deductive systems, or as experimental constructs in situations where it is too dangerous or difficult to experiment on the primary system, or where the primary system is too complex to model mathematically.[1 pp 50-63] None of the perpetrators of public virus outbreaks have utilized models in their work, indicating that they are, in fact, not pursuing knowledge in an acceptable fashion and indicating that a defense of 'particular ignorance' is technically indefensible.

Conclusion

The ethical considerations presented indicate several areas in which educational and scientific establishments can and should take action. It is necessary to offer within the scientific and technological curricula a forum for the conceptualization of the creative process and a presentation and discussion of the attendant ethical concerns. An inter-disciplinary examination of technological developments and their probable impact needs to have wider discussion as a way of breaking the isolation of the scientific process.

It is the responsibility of the university faculties and information managers to instruct and support students and researchers in the proper development of protocols which model responsible experimental behavior. Likewise, management should seek to provide the opportunity to test experimental constructs within a system both closed and yet large enough to render significant results.

The question of accountability, particularly within the distinction between vincible and invincible ignorance must be discussed. Those who attack the infrastructure of society cannot do so with impunity under the clock of scientific license. Again, universities and research institutions need to develop and make clear the acceptable parameters of experimentation and sanctions to be taken in response to the abrogation of these guidelines.

[1] Barbour, Ian. Science and Secularity: The Ethics of Technology. Harper and Row, Publishers, New York 1970.

[2] Baumod, William. Superfairness: Applications and Theory. The MIT Press, Cambridge. 1987.

[3] Campbell, Douglas, "Computer Contagion",Security Management, Vol 32 Issue 10, Oct 1988, pp 83-83.

[4] Cohen, Fred," Computer Viruses: Theory and Experiments," Computers and Security, Volume 6 Issue 1, Feb 1987

[5]. Connolly, James; Alexander, Michael. "Worm Impact Outlives System Threat: Managers Prepare for CEO Grilling; Security, Ethics Under National Scrutiny", Computerworld, Vol 22, Issue 46, Nov. 14, 1988.

[6] Gibson. Steve ,"What Were Simple Viruses May Fast Become a Plague,"InfoWorld , Vol 10 Issue 18, May 2, 1988

[7] Jonas, Hans. The Imperative of Responsibility: In Search of Ethics for the Technological Age. The University of Chicago Press. Chicago. 1984.

[8] Kask, Alex ,"Confidentiality of Data Is at Greater Risk in the Age of Laptops and Viruses,"InfoWorld, Vol 10 Issue 16, Apr 18.

[9] Murray, W.H. "The Application of Epidemiology to Computer Viruses," Computers and Security, Vol 7 Issue 2 Apr 1988

[10] Cssowska, Maria. Social Determinants of Moral Ideas. University of Pennsylvania Press. Philadelphia. 1970.

[11]. Rosenberg, Robert. "Sick Software? Network Virus? Insurance Won't Help." Data Communications, Vol. 17, Issue 6, June 1988.

[12]. Rosentahl, Beth Ellyn. "Computer Viruses Can Make Your Bank Sick," Bankers Monthly, Vol. 105, Issue 10, Oct. 1988.

[13] Singer, Peter. Practical Ethics. Cambridge University Press. Cambridge. 1979.

Concerning Hackers Who Break into Computer Systems

Dorothy E. Denning
Digital Equipment Corp., Systems Research Center
130 Lytton Ave., Palo Alto, CA 94301
415-853-2252, denning@src.dec.com

Abstract

A diffuse group of people, often called "hackers," has been characterized as unethical, irresponsible, and a serious danger to society for actions related to breaking into computer systems. This paper attempts to construct a picture of hackers, their concerns, and the discourse in which hacking takes place. My initial findings suggest that hackers are learners and explorers who want to help rather than cause damage, and who often have very high standards of behavior. My findings also suggest that the discourse surrounding hacking belongs at the very least to the gray areas between larger conflicts that we are experiencing at every level of society and business in an information age where many are not computer literate. These conflicts are between the idea that information cannot be owned and the idea that it can, and between law enforcement and the First and Fourth Amendments. Hackers have raised important issues about values and practices in an information society. Based on my findings, I recommend that we work closely with hackers, and suggest several actions that might be taken.

1 Introduction

The world is crisscrossed with many different networks that are used to deliver essential services and basic necessities – electric power, water, fuel, food, goods, to name a few. These networks are all publicly accessible and hence vulnerable to attacks, and yet virtually no attacks or disruptions actually occur.

The world of computer networking seems to be an anomaly in the firmament of networks. Stories about attacks, breakins, disruptions, theft of information, modification of files, and the like appear frequently in the newspapers. A diffuse group called "hackers" is often the target of scorn and blame for these actions. Why are computer networks any different from other vulnerable public networks? Is the difference the result of growing pains in a young field? Or is it the reflection of deeper tensions in our emerging information society?

There are no easy or immediate answers to these questions. Yet it is important to our future in a networked, information-dependent world that we come to grips with them. I am deeply interested in them. This paper is my report of what I have discovered in the early stages of what promises to be a longer investigation. I have concentrated my attention in these early stages on the hackers themselves. Who are they? What do they say? What motivates them? What are their values? What do that have to say about public policies regarding information and computers? What do they have to say about computer security?

From such a profile I expect to be able to construct a picture of the discourses in which hacking takes place. By a discourse I mean the invisible background of assumptions that transcends individuals and governs our ways of thinking, speaking, and acting. My initial findings lead me to conclude that this discourse belongs at the very least to the gray areas between larger conflicts that we are experiencing at every level of society and business, the conflict between the idea that information cannot be owned and the idea that it can, and the conflict between law enforcement and the First and Fourth Amendments.

But, enough of the philosophy. On with the story!

2 Opening Moves

In late fall of 1989, Frank Drake (not his real name), editor of the now defunct cyberpunk magazine W.O.R.M., invited me to be interviewed for the magazine. In accepting the invitation, I hoped that something I might say would discourage hackers from breaking into systems. I was also curious about the hacker-culture. This seemed like a good opportunity to learn about it.

The interview was conducted electronically. I quickly discovered that I had much more to learn from Drake's questions than to teach. For example, he asked: "Is providing computer security for large databases that collect information on us a real service? How do you balance the individual's privacy vs. the corporations?" This question surprised me. Nothing that I had read about hackers ever suggested that they might care about privacy. He also asked: "What has (the DES) taught us about what the government's (especially NSA's) role in cryptography should be?" Again, I was surprised to discover a concern for the role of the government in computer security. I did not know at the time that I would later discover considerable overlap in the issues discussed by hackers and those of other computer professionals.

I met with Drake to discuss his questions and views. After our meeting, we continued our dialog electronically with me interviewing him. This gave me the opportunity to explore his views in greater depth. Both interviews appear in *Computers Under Attack*, edited by Peter Denning [6].

My dialog with Drake increased my curiosity about hackers. I read articles and books by or about hackers. In addition, I had discussions with nine hackers whom I will not mention by name. Their ages ranged from 17 to 28.

The word "hacker" has taken on many different meanings ranging from 1) "a person who enjoys learning the details of computer systems and how to stretch their capabilities" to 2) "a malicious or inquisitive meddler who tries to discover information by poking around ... possibly by deceptive or illegal means ..." [33]. The hackers described in this paper are both learners and explorers who sometimes perform illegal actions. However, all of the hackers I spoke with said they did not engage in or approve of malicious acts that damage systems or files. Thus, this paper is not about malicious hackers. Indeed, my research so far suggests that there are very few malicious hackers. Neither is this paper about career criminals who, for example, defraud businesses, or about people who use stolen credit cards to purchase goods. The characteristics of many of the hackers I am writing about are summed up in the words of one of the hackers: "A hacker is someone who experiments with systems... (Hacking) is playing with systems and making them do what they were never intended to do. Breaking in and making free calls is just a small part of that. Hacking is also about freedom of speech and free access to information – being able to find out anything. There is also the David and Goliath side of it, the underdog vs. the system, and the ethic of being a folk hero, albeit a minor one."

Richard Stallman, founder of the Free Software Foundation who calls himself a hacker according to the first sense of the word above, recommends calling security-breaking hackers "crackers" [31]. While this description may be more accurate, I shall use the term "hacker" since the people I am writing about call themselves hackers and all are interested in learning about computer and communication systems. However, there are many people like Stallman who call themselves hackers and do not engage in illegal or deceptive practices; this paper is also not about those hackers.

In what follows I will report on what I have learned about hackers from hackers. I will organize the discussion around the principal domains of concerns I observed. I recommend Meyer's thesis [26] for a more detailed treatment of the hackers' social culture and networks, and Meyer and Thomas [27] for an interesting interpretation of the computer underground as a postmodernist rejection of conventional culture that substitutes "rational technological control of the present for an anarchic and playful future."

I do not pretend to know all the concerns that hackers have, nor do I claim to have conducted a scientific study. Rather, I hope that my own informal study motivates others to explore the area further. It is essential that we as computer security professionals take into account hackers' concerns in the design of our policies, procedures, laws regulating computer and information access, and educational programs. Although I speak about security-breaking hackers as a group, their competencies, actions, and views are not all the same. Thus, it is equally important that our policies and programs take into account individual differences.

In focusing on what hackers say and do, I do not mean for a moment to set aside the concerns of the owners and users of systems that hackers break into, the concerns of law enforcement personnel, or our own concerns as computer security professionals. But I do recommend that we work closely with hackers as well as these other groups to design new approaches and programs for addressing the concerns of all. Like ham radio operators, hackers exist, and it is in our best interest that we learn to communicate and work with them rather than against them.

I will suggest some actions that we might consider taking, and I invite others to reflect on these and suggest their own. Many of these suggestions are from the hackers themselves; others came from the recommendations of the ACM Panel on Hacking [20] and from colleagues.

I grouped the hackers' concerns into five categories: access to computers and information for learning; thrill, excitement and challenge; ethics and avoiding damage; public image and treatment; and privacy and first amendment rights. These are discussed in the next five subsections. I have made an effort to present my findings as uncritical observations. The reader should not infer that I either approve or disapprove of actions hackers take.

3 Access to Computers and Information for Learning

Although Levy's book *Hackers* [21] is not about today's security-breaking hackers, it articulates and interprets a "hacker ethic" that is shared by many of these hackers. The ethic includes two key principles that were formulated in the early days of the AI Lab at MIT: "Access to computers – and anything which might teach you something about the way the world works – should be unlimited and total," and "All information should be free." In the context in which these principles were formulated, the computers of interest were research machines and the information was software and systems information.

Since Stallman is a leading advocate of open systems and freedom of information, especially software, I asked him what he means by this. He said: "I believe that all generally useful information should be free. By 'free' I am not referring to price, but rather to the freedom to copy the information and to adapt it to one's own uses." By "generally useful" he does not include confidential information about individuals or credit card information, for example. He further writes: "When information is generally useful, redistributing it makes humanity wealthier no matter who is distributing and no matter who is receiving." Stallman has argued strongly against user interface copyright, claiming that it does not serve the users or promote the evolutionary process [32].

I asked hackers whether all systems should be accessible and all information should be free. They said that it is OK if some systems are closed and some information, mainly confidential information about individuals, is not accessible. They make a distinction between information about security technology and confidential information protected by that technology, arguing that it is the former that should be accessible. They said that information hoarding is inefficient and slows down evolution of technology. They also said that more systems should be open so that idle resources are not wasted. One hacker said that the high costs of communication hurts the growth of the information economy.

These views of information sharing seem to go back at least as far as the 17th and 18th centuries. Samuelson [29] notes that "The drafters of the Constitution, educated in the Enlightenment tradition, shared that era's legacy of faith in the enabling powers of knowledge for society as well as the individual." She writes that our current copyright laws, which protect the expression of information, but not the information itself, are based on the belief that unfettered and widespread dissemination of information promotes technological progress. (Similarly for patent laws which protect devices and processes, not the information about them.) She cites two recent court cases where courts reversed the historical trend and treated information as ownable property. She raises questions about whether in entering the Information Age where information is the source of greatest wealth, we have outgrown the Enlightenment tradition and are coming to treat information as property.

In a society where knowledge is said to be power, Drake expressed particular concern about what he sees as a growing information gap between the rich and poor. He would like to see information that is not about individuals be made public, although it could still be owned. He likes to think that companies would actually find it to their advantage to share information. He noted how IBM's disclosure of the PC allowed developers to make more products for the computers, and how Adobe's disclosure of their font representation helped them compete against the proposed Apple-Microsoft collaboration. He recognizes that in our current political framework, it is difficult to make all information public, because complicated structures have been built on top of an assumption that certain information will be kept secret. He cites our defense policy, which is founded on secrecy for military information, as an example.

Hackers say they want access to information and computing and network resources in order to learn. Both Levy [21] and Landreth [19] note that hackers have an intense, compelling interest in computers and learning, and many go into computers as a profession. Some hackers break into systems in order to learn more about how the systems work. Landreth says these hackers want to remain undiscovered so that they can stay on the system as long as possible. Some of them devote most of their time to learning how to break the locks and other security mechanisms on systems; their background in systems and programming varies considerably. One hacker wrote "A hacker sees a security hole and takes advantage of it because it is there, not to destroy information or steal. I think our activities would be analogous to someone discovering methods of acquiring information in a library and becoming excited and perhaps engrossed."

We should not underestimate the effectiveness of the networks in which hackers learn their craft. They do research, learn about systems, work in groups, write, and teach others. One hacker said that he belongs to a study group with the mission of churning out files of information and learning as much as possible. Within the group, people specialize, collaborate on research projects, share information and news, write articles, and teach others about their areas of specialization. Hackers have set up a private system of education that engages them, teaches them to think, and allows them to apply their knowledge in purposeful, if not always legal, activity. Ironically, many of our nation's classrooms have been criticized for providing a poor learning environment that seems to emphasize memorization rather than thinking and reasoning. One hacker reported that through volunteer work with a local high school, he was trying to get students turned on to learning.

Many hackers say that the legitimate computer access they have through their home and school computers do not meet their needs. One student told me that his high school did not offer anything beyond elementary courses in BASIC and PASCAL, and that he was bored by these. Hans Huebner, a hacker in Germany who goes by the name Pengo, wrote in a note to the RISKS Forum [18] : "I was just interested in computers, not in the data which has been kept on their disks. As I was going to school at that time, I didn't even have the money to buy my own computer. Since CP/M (which was the most sophisticated OS I could use on machines which I had legal access to) didn't turn me on anymore, I enjoyed the lax security of the systems I had access to by using X.25 networks. You might point out that I should have been patient and waited until I could go to the university and use their machines. Some of you might understand that waiting was just not the thing I was keen on in those days."

Brian Harvey, in his position paper [16] for the ACM Panel on Hacking, claims that the computer medium available to students, such as BASIC and floppy disks, is inadequate for challenging intellectual work. His recommendation is that students be given access to real computing power, and that they be taught how to use that power responsibly. He describes a program he created at a public high school in Massachusetts during the period 1979-1982. They installed a PDP-11/70 and let students and teachers carry out the administration of the system. Harvey assessed that putting the burden of dealing with the problems of malicious users on the students themselves was a powerful educational force. He also noted that the students who had the skill and interest to be password hackers were discouraged from this activity because they also wanted to keep the trust of their colleagues in order that they could acquire "superuser" status on the system.

Harvey also makes an interesting analogy between teaching computing and teaching karate. In karate instruction, students are introduced to the real, adult community. They are given access to a powerful, deadly weapon, and at the same time are taught discipline and responsibility. Harvey speculates that the reason that students do not misuse their power is that they know they are being trusted with something important, and they want to live up to that trust. Harvey applied this principle when he set up the school system.

The ACM panel endorsed Harvey's recommendation, proposing a three-tiered computing environment with local, district-wide, and nation-wide networks. They recommended that computer professionals participate in this effort as mentors and role models. They also recommended that government and industry be encouraged to establish regional computing centers using donated or re-cycled equipment; that students be apprenticed to local companies either part-time on a continuing basis or on a periodic basis; and, following a suggestion from Felsenstein [9] for a "Hacker's League," that a league analogous to the Amateur Radio Relay League be established to make contributed resources available for educational purposes.

Drake said he liked these recommendations. He said that if hackers were given access to powerful systems through a public account system, they would supervise themselves. He also suggested that Computer Resource Centers be established in low-income areas in order to help the poor get access to information. Perhaps hackers could help run the centers and teach the members of the community how to use the facilities. One of my colleagues suggested cynically that the hackers would only use this to teach the poor how to hack rich people's systems. A hacker responded by saying this was ridiculous; hackers would not teach people how to break into systems, but rather how to use computers effectively and not be afraid of them. In addition, the hackers I spoke with who had given up illegal activities said they stopped doing so when they got engaged in other work.

Geoff Goodfellow and Richard Stallman have reported that they have given hackers accounts on systems that they manage, and that the hackers have not misused the trust granted to them. Perhaps universities could consider providing accounts to pre-college students on the basis of recommendations from their teachers or parents. The students might be challenged to work on the same homework problems assigned in courses or to explore their own interests. Students who strongly dislike the inflexibility of classroom learning might excel in an environment that allows them to learn on their own, in much the way that hackers have done.

4 Thrill, Excitement, and Challenge

One hacker wrote that "Hackers understand something basic about computers, and that is that they can be enjoyed. I know none who hack for money, or hack to frighten the company, or hack for anything but fun."

In the words of another hacker, "Hacking was the ultimate cerebral buzz for me. I would come home from another dull day at school, turn my computer on, and become a member of the hacker elite. It was a whole different world where there were no condescending adults and you were judged only by your talent. I would first check in to the private Bulletin Boards where other people who were like me would hang out, see what the news was in the community, and trade some info with people across the country. Then I would start actually hacking. My brain would be going a million miles an hour and I'd basically completely forget about my body as I would jump from one computer to another trying to find a path into my target. It was the rush of working on a puzzle coupled with the high of discovery many magnitudes intensified. To go along with the adrenaline rush was the illicit thrill of doing something illegal. Every step I made could be the one that would bring the authorities crashing down on me. I was on the edge of technology and exploring past it, spelunking into electronic caves where I wasn't supposed to be."

The other hackers I spoke with made similar statements about the fun and challenge of hacking. In SPIN magazine [7], reporter Julian Dibbell speculated that much of the thrill comes from the dangers associated with the activity, writing that "the technology just lends itself to cloak-and-dagger drama," and that "hackers were already living in a world in which covert action was nothing more than a game children played."

Eric Corley [3] characterizes hacking as an evolved form of mountain climbing. In describing an effort to construct a list of active mailboxes on a Voice Messaging System, he writes "I suppose the main reason I'm wasting my time pushing all these buttons is simply so that I can make a list of something that I'm not supposed to have and be the first person to accomplish this." He said that he was not interested in obtaining an account of his own on the system. Gordon Meyer says he found this to be a recurring theme: "We aren't supposed to be able to do this, but we can" – so they do.

One hacker said he was now working on anti-viral programming. He said it was almost as much fun as breaking into systems, and that it was an intellectual battle against the virus author.

5 Ethics and Avoiding Damage

All of the hackers I spoke with said that malicious hacking was morally wrong. They said that most hackers are not intentionally malicious, and that they themselves are concerned about causing accidental damage. When I asked Drake about the responsibility of a person with a PC and modem, his reply included not erasing or modifying anyone else's data, and not causing a legitimate user on a system any problems. Hackers say they are outraged when other hackers cause damage or use resources that would be missed, even if the results are unintentional and due to incompetence. One hacker wrote "I have *always* strived to do *no* damage, and to inconvenience as few people as possible. *I never, ever, ever delete a file.* One of the first commands I do on a new system is disable the delete file command." Some hackers say that it is unethical to give passwords and similar security-related information to persons who might do damage. In the recent incident where a hacker broke into Bell South and downloaded a text file on the emergency 911 service, hackers say that there was no intention to use this knowledge to break into or sabotage the 911 system. According to Emmanuel Goldstein [12], the file did not even contain information about how to break into the 911 system.

The hackers also said that some break-ins were unethical, such as breaking into hospital systems, and that it is wrong to read confidential information about individuals or steal classified information. All said it was wrong to commit fraud for personal profit.

Although we as computer security professionals often disagree with hackers about what constitutes damage, the ethical standards listed here sound much like our own. Where the hackers' ethics differ from the standards adopted by most in the computer security community is that hackers say it is not unethical to break into many systems, use idle computer and communications resources, and download system files in order to learn. Goldstein says that hacking is not wrong: it is not the same as stealing, and uncovers design flaws and security deficiencies [11].

Brian Reid, a colleague at Digital who has spoken with many hackers, speculates that a hacker's ethics may come from not being raised properly as a civilized member of society, and not appreciating the rules of living in society. One hacker responded to this with "What does 'being brought up properly' mean? Some would say that it is 'good' to keep to yourself, mind your own business. Others might argue that it is healthy to explore, take risks, be curious and discover." Brian Harvey [16] notes that many hackers are adolescents, and that adolescents are at a less advanced stage of moral development than adults, where they might not see how the effects of their actions hurt others. Larry Martin [25] claims that parents, teachers, the press, and others in society are not aware of their responsibility to contribute to instilling ethical values associated with computer use. This could be the consequence of the youth of the computing field; many people are still computer illiterate and cultural norms may be lagging behind advances in technology and the growing dependency on that technology by businesses and society. Hollinger and Lanza-Kaduce [17] speculate that the cultural normative messages about the use and abuse of computer technology have been driven by the adoption of criminal laws in the last decade. They also speculate that hacking may be encouraged during the process of becoming computer literate. Some of my colleagues say that hackers are irresponsible. One hacker responded "I think it's a strong indication of the amount of responsibility shown that so *few* actually *damaging* incidents are known."

But we must not overlook that the differences in ethics also reflect a difference in philosophy about information and information handling resources; whereas hackers advocate sharing, we seem to be advocating ownership as property. The differences also represent an opportunity to examine our own ethical behavior and our practices for information sharing and protection. For example, one hacker wrote "I will accept that it is morally wrong to copy some proprietary software, however, I think that it is morally wrong to charge $6000 for a program that is only around 25K long." Hence, I shall go into a few of the ethical points raised by hackers more closely. It is not a simple case of good or mature (us) against bad or immature (hackers), or of teaching hackers a list of rules.

Many computer professionals such as Martin [25] argue the moral questions by analogy. The analogies are then used to justify their judgment of a hacker's actions as unethical. Breaking into a system is compared with breaking into a house, and downloading information and using computer and telecommunications services is compared with stealing tangible goods. But, say hackers, the situations are not the same. When someone breaks into a house, the objective is to steal goods, which are often irreplaceable, and property is often damaged in the process. By contrast, when a hacker breaks into a system, the objective is to learn and avoid causing damage. Downloaded information is copied, not stolen, and still exists on the original system. Moreover, as noted earlier, information has not been traditionally regarded as property. Dibbel [7] says that when the software industries and phone companies claim losses of billions of dollars to piracy, they are not talking about goods that disappear from the shelves and could have been sold.

We often say that breaking into a system implies a lack of caring for the system's owner and authorized users. But, one hacker says that the ease of breaking into a system reveals a lack of caring on the part of the system manager to protect user and company assets, or failure on the part of vendors to warn managers about the vulnerabilities of their systems. He estimated his success rate of breaking in at 10-15%, and that is without spending more than an hour on any one target system. Another hacker says that he sees messages from vendors notifying the managers, but that the managers fail to take action.

Richard Pethia of CERT (Computer Emergency Response Team) reports that they seldom see cases of malicious damage caused by hackers, but that the break-ins are nevertheless disruptive because system users and administrators want to be sure that nothing was damaged. (CERT suggests that sites reload system software from secure backups and change all user passwords in order to protect against possible back doors and Trojan Horses that might have been planted by the hacker. Pethia also noted that prosecutors are generally called for government sites, and are being called for non-government sites with increasing frequency.) Pethia says that break-ins also generate a loss of trust in the computing environment, and may lead to adoption of new policies that are formulated in a panic or management edicts that severely restrict connectivity to outside systems. Brian Harvey says that hackers cause damage by increasing the amount of paranoia, which in turn leads to tighter security controls that diminish the quality of life for the users. Hackers respond to these points by saying they are the scapegoats for systems that are not adequately protected. They say that the paranoia is generated by ill-founded fears and media distortions (I will return to this point later), and that security need not be oppressive to keep hackers out; it is mainly making sure that passwords and system defaults are well chosen.

Pethia says that some intruders seem to be disruptive to prove a point, such as that the systems are vulnerable, the security personnel are incompetent, or "it's not nice to say bad things about hackers." In the N.Y. Times, John Markoff [24] wrote that the hacker who claimed to have broken into Cliff Stoll's system said he was upset by Stoll's portrayal of hackers in *The Cuckoo's Egg* [34]. Markoff reported that the caller said: "He (Stoll) was going on about how he hates all hackers, and he gave pretty much of a one-sided view of who hackers are."

The Cuckoo's Egg captures many of the popular stereotypes of hackers. Criminologist Jim Thomas criticizes it for presenting a simplified view of the world, one where everything springs from the forces of light (us) or of darkness (hackers) [35]. He claims that Stoll fails to see the similarities between his own activities (e.g., monitoring communications, "borrowing" monitors without authorization, shutting off network access without warning, and lying to get information he wants) and those of hackers. He points out Stoll's use of pejorative words such as "varmint" to describe hackers, and Stoll's quote of a colleague: "They're technically skilled but ethically bankrupt programmers without any respect for others' work – or privacy. They're not destroying one or two programs. They're trying to wreck the cooperation that builds our networks," [34, pages 159]. Thomas writes "at an intellectual level, it (Stoll's book) provides a persuasive, but simplistic, moral imagery of the nature of right and wrong, and provides what – to a lay reader – would seem a compelling justification for more statutes and severe penalties against the computer underground. This is troublesome for two reasons. First, it leads to a mentality of social control by law enforcement during a social phase when some would argue we are already over-controlled. Second, it invokes a punishment model that assumes we can stamp out behaviors to which we object if only we apprehend and convict a sufficient number of violators. ... There is little evidence that punishment will in the long run reduce any given offense, and the research of Gordon Meyer and I suggests that criminalization may, in fact, contribute to the growth of the computer underground."

6 Public Image and Treatment

Hackers express concern about their negative public image and identity. As noted earlier, hackers are often portrayed as being irresponsible and immoral. One hacker said that "government propaganda is spreading an image of our being at best, sub-human, depraved, criminally inclined, morally corrupt, low life. We need to prove that the activities that we are accused of (crashing systems, interfering with life support equipment, robbing banks, and jamming 911 lines) are as morally abhorrent to us as they are to the general public."

The public identity of an individual or group is generated in part by the actions of the group interacting with the standards of the community observing those actions. What then accounts for the difference between the hacker's public image and what they say about themselves? One explanation may be the different standards. Outside the hacking community, the simple act of breaking into systems is regarded as unethical by many. The use of pejorative words like "vandal" and "varmint" reflect this discrepency in ethics. Even the word "criminal" carries with it connotations of someone evil; hackers say they are not criminal in this sense. Katie Hafner notes that Robert Morris, who was convicted of launching the Internet worm, was likened to a terrorist even though the worm did not destroy data [14].

Distortions of events and references to potential threats also create an image of persons who are dangerous. Regarding the 911 incident where a hacker downloaded a file from Bell South, Goldstein reported "Quickly, headlines screamed that hackers had broken into the 911 system and were interfering with emergency telephone calls to the police. One newspaper report said there were no indications that anyone had died or been injured as a result of the intrusions. What a relief. Too bad it wasn't true," [12]. In fact, the hackers involved with the 911 text file had not broken into the 911 system. The dollar losses attributed to hacking incidents also are often highly inflated.

Thomas and Meyer [36] say that the rhetoric depicting hackers as a dangerous evil contributes to a "witch hunt" mentality, wherein a group is first labeled as dangerous, and then enforcement agents are mobilized to exorcise the alleged social evil. They see the current sweeps against hackers as part of a reaction to a broader fear of change, rather than to the actual crimes committed.

Hackers say they are particularly concerned that computer security professionals and system managers do not appear to understand hackers or be interested in their concerns. Hackers say that system managers treat them like enemies and criminals, rather than as potential helpers in their task of making their systems secure. This may reflect managers' fears about hackers, as well as their responsibilities to protect the information on their systems. Stallman says that the strangers he encounters using his account are more likely to have a chip on their shoulder than in the past; he attributes this to a harsh enforcer mentality adopted by the establishment. He says that network system managers start out with too little trust and a hostile attitude toward strangers that few of the strangers deserve. One hacker said that system managers show a lack of openness to those who want to learn.

Stallman also says that the laws make the hacker scared to communicate with anyone even slightly "official," because that person might try to track the hacker down and have him or her arrested. Drake raised the issue of whether the laws could differentiate between malicious and nonmalicious hacking, in support of a "kinder, gentler" relationship between hackers and computer security people. In fact, many states such as California initially passed computer crime laws that excluded malicious hacking; it was only later that these laws were amended to include nonmalicious actions [17]. Hollinger and Lanza-Kaduce speculate that these amendments and other new laws were catalyzed mainly by media events, especially the reports on the "414 hackers" and the movie "War Games," which created a perception of hacking as extremely dangerous, even if that perception was not based on facts.

Hackers say they want to help system managers make their systems more secure. They would like managers to recognize and use their knowledge about system vulnerabilities. Landreth [19] suggests ways in which system managers can approach hackers in order to turn them into colleagues, and Goodfellow also suggests befriending hackers [13]. John Draper (Cap'n Crunch) says it would help if system managers and the operators of phone companies and switches could cooperate in tracing a hacker without bringing in law enforcement authorities.

Drake suggests giving hackers free access in exchange for helping with security, a suggestion that I also heard from several hackers. Drake says that the current attitude of treating hackers as enemies is not very conducive to a solution, and by belittling them, we only cause ourselves problems.

I asked some of the hackers whether they'd be interested in breaking into systems if the rules of the "game" were changed so that instead of being threatened by prosecution, they were invited to leave a "calling card" giving their name, phone number, and method of breaking in. In exchange, they would get recognition and points for each vulnerability they discovered. Most were interested in playing; one hacker said he would prefer monetary reward since he was supporting himself. Any system manager interested in trying this out could post a welcome message inviting hackers to leave their cards. This approach could have the advantage of not only letting the hackers contribute to the security of the system, but of allowing the managers to quickly recognize the potentially malicious hackers, since they are unlikely to leave their cards. Perhaps if hackers are given the opportunity to make contributions outside the underground, this will dampen their desire to pursue illegal activities.

Several hackers said that they would like to be able to pursue their activities legally and for income. They like breaking into systems, doing research on computer security, and figuring out how to protect against vulnerabilities. They say they would like to be in a position where they have permission to hack systems. Goodfellow suggests hiring hackers to work on tiger teams that are commissioned to locate vulnerabilities in systems through penetration testing. Baird Info-Systems Safeguards, Inc., a security consulting firm, reports that they have employed hackers on several assignments [1]. They say the hackers did not violate their trust or the trust of their clients, and performed in an outstanding manner. Baird believes that system vulnerabilities can be better identified by employing people who have exploited systems.

One hacker suggested setting up a clearinghouse that would match hackers with companies that could use their expertise, while maintaining anonymity of the hackers and ensuring confidentiality of all records. Another hacker, in describing an incident where he discovered a privileged account without a password, said "What I (and others) wish for is a way that hackers can give information like this to a responsible source, *and have hackers given credit for helping*! As it is, if someone told them that 'I'm a hacker, and I *really* think you should know...' they would freak out, and run screaming to the SS (Secret Service) or the FBI. Eventually, the person who found it would be caught, and hauled away on some crazy charge. If they could only just *accept* that the hacker was trying to help!" The clearinghouse could also provide this type of service.

Hackers are also interested in security policy issues. Drake expressed concern over how we handle information about computer security vulnerabilities. He argues that it is better to make this information public than cover it up and pretend that it does not exist, and cites the CERT to illustrate how this approach can be workable. Other hackers, however, argue for restricting initial dissemination of flaws to customers and users. Drake also expressed concern about the role of the government, particularly the military, in cryptography. He argues that NSA's opinion on a cryptographic standard should be taken with a large grain of salt because of their code breaking role.

Some security specialists are opposed to hiring hackers for security work, and Eugene Spafford has urged people not to do business with any company that hires a convicted hacker to work in the security area [22]. He says that "This is like having a known arsonist install a fire alarm." But, the laws are such that a person can be convicted for having done nothing other than break into a system; no serious damage (i.e., no "computer arson") is necessary. Many of our colleagues, including Geoff Goodfellow [13] and Brian Reid [10], admit to having broken into systems in the past. Reid is quoted as saying that because of the knowledge he gained breaking into systems as a kid, he was frequently called in to help catch people who break in. Spafford says that times have changed, and that this method of entering the field is no longer socially acceptable, and fails to provide adequate training in computer science and computer engineering [30]. However, from what I have observed, many hackers do have considerable knowledge about telecommunications, data security, operating systems, programming languages, networks, and cryptography. But, I am not challenging a policy to hire competent people of sound character. Rather, I am challenging a strict policy that uses economic pressure to close a field of activity to all persons convicted of breaking into systems. It is enough that a company is responsible for the behavior of its employees. Each hacker can be considered for employment based on his or her own competency and character.

Some people have called for stricter penalties for hackers, including prison terms, in order to send a strong deterrent message to hackers. John Draper, who was incarcerated for his activities in the 1970's, argues that in practice this will only make the problem worse. He told me that he was forced under threat to teach other inmates his knowledge of communications systems. He believes that prison sentences will serve only to spread hacker's knowledge to career criminals. He said he was never approached by criminals outside the prison, but that inside the prison they had control over him.

One hacker said that by clamping down on the hobbyist underground, we will only be left with the criminal underground. He said that without hackers to uncover system vulnerabilities, the holes will be left undiscovered, to be utilized by those likely to cause real damage.

Goldstein argues that the existing penalties are already way out of proportion to the acts committed, and that the reason is because of computers [11]. He says that if Kevin Mitnick had committed crimes similar to those he committed but without a computer, he would have been classified as a mischief maker and maybe fined $100 for trespassing; instead, he was put in jail without bail [11]. Craig Neidorf, a publisher and editor of the electronic newsletter *Phrack*, faces up to 31 years and a fine of $122,000 for receiving, editing, and transmitting the downloaded text file on the 911 system [12].

7 Privacy and the First and Fourth Amendments

The hackers I spoke with advocated privacy protection for sensitive information about individuals. They said they are not interested in invading people's privacy, and that they limited their hacking activities to acquiring information about computer systems or how to break into them. There are, of course, hackers who break into systems such as the TRW credit database. Emanuel Goldstein argues that such invasions of privacy took place before the hacker arrived [15]. Referring to credit reports, government files, motor vehicle records, and the "megabytes of data piling up about each of us," he says that thousands of people legally can see and use this data, much of it erroneous. He claims that the public has been misinformed about the databases, and that hackers have become scapegoats for the holes in the systems. One hacker questioned the practice of storing sensitive personal information on open systems with dial-up access, the accrual of the information, the methods used to acquire it, and the purposes to which it is put. Another hacker questioned the ethics of including religion and race in credit records.

Drake told me that he was concerned about the increasing amount of information about individuals that is stored in large data banks, and the inability of the individual to have much control over the use of that information. He suggests that the individual might be co-owner of information collected about him or her, with control over the use of that information. He also says that an individual should be free to withhold personal information, of course paying the consequences of doing so (e.g., not getting a drivers license or credit card). In fact, all Federal Government forms are required to contain a Privacy Act Statement that states how the information being collected will be used and, in some cases, giving the option of withholding the information.

Goldstein has also challenged the practices of law enforcement agencies in their attempt to crack down on hackers [12]. He said that all incoming and outgoing electronic mail used by *Phrack* was monitored before the newsletter was shutdown by authorities. "Had a printed magazine been shut down in this fashion after having all of their mail opened and read, even the most thick-headed sensationalist media types would have caught on: hey, isn't that a violation of the First Amendment?" He also cites the shutdown of several bulletin boards as part of Operation Sun Devil, and quotes the administrator of the bulletin board Zygot as saying "Should I start reading my users' mail to make sure they aren't saying anything naughty? Should I snoop through all the files to make sure everyone is being good? This whole affair is rather chilling." The administrator for the public system The Point wrote "Today, there is no law or precedent which affords me ... the same legal rights that other common carriers have against prosecution should some other party (you) use my property (The Point) for illegal activities. That worries me ..."

About 40 personal computer systems and 23,000 data disks were seized under Operation Sun Devil, a two-year investigation involving the FBI, Secret Service, and other federal and local law enforcement officials. In addition, the Secret Service acknowledges that its agents, acting as legitimate users, had secretly monitored computer bulletin boards [23]. Markoff reports that California Representative Don Edwards, industry leader Mitchell Kapor, and civil liberties advocates are alarmed by these government actions, saying that they challenge freedom of speech under the First Amendment and protection against searches and seizures under the Fourth Amendment. Markoff asks: "Will fear of hackers bring oppression?"

John Barlow writes "The Secret Service may actually have done a service for those of us who love liberty. They have provided us with a devil. And devils, among their other galvanizing virtues, are just great for clarifying the issues and putting iron in your spine," [2]. Some of the questions that Barlow says need to be addressed include "What are data and what is free speech? How does one treat property which has no physical form and can be infinitely reproduced? Is a computer the same as a printing press?" Barlow urges those of us who understand the technology to address these questions, lest the answers be given to us by law makers and law enforcers who do not. Barlow and Kapor are constituting a foundation to "raise and disburse funds for education, lobbying, and litigation in the areas relating to digital speech and the extension of the Constitution into Cyberspace."

8 Conclusions

Hackers say that it is our social responsibility to share information, and that it is information hoarding and disinformation that are the crimes. This ethic of resource and information sharing contrasts sharply with computer security policies that are based on authorization and "need to know." This discrepancy raises an interesting question: Does the hacker ethic reflect a growing force in society that stands for greater sharing of resources and information – a reaffirmation of basic values in our constitution and laws? It is important that we examine the differences between the standards of hackers, systems managers, users, and the public. These differences may represent breakdowns in current practices, and may present new opportunities to design better policies and mechanisms for making computer resources and information more widely available.

The sentiment for greater information sharing is not restricted to hackers. In the best seller, *Thriving on Chaos*, Tom Peters [28, pages 610] writes about sharing within organizations: "Information hoarding, especially by politically motivated, power-seeking staffs, has been commonplace throughout American industry, service and manufacturing alike. It will be an impossible millstone around the neck of tomorrow's organizations. Sharing is a must." Peters argues that information flow and sharing is fundamental to innovation and competitiveness. On a broader scale, Peter Drucker [8] says that the "control of information by government is no longer possible. Indeed, information is now transnational. Like money, it has no 'fatherland.' "

Nor is the sentiment restricted to people outside the computer security field. Harry DeMaio [4] says that our natural urge is to share information, and that we are suspicious of organizations and individuals who are secretive. He says that information is exchanged out of "want to know" and mutual accommodation rather than "need to know." If this is so, then some of our security policies are out of step with the way people work. Peter Denning [5] says that information sharing will be widespread in the emerging worldwide networks of computers and that we need to focus on "immune systems" that protect against mistakes in our designs and recover from damage.

I began my investigation of hackers with the question, who are they and what is their culture and discourse? My investigation uncovered some of their concerns, which provided the organizational structure to this paper, and several suggestions for new actions that might be taken. My investigation also opened up a broader question: What conflict in society do hackers stand at the battle lines of? Is it owning or restricting information vs. sharing information – a tension between an age-old tradition of controlling information as property and the Englightenment tradition of sharing and disseminating information? Is it controlling access based on "need to know," as determined by the information provider, vs. "want to know," as determined by the person desiring access? Is it law enforcement vs. freedoms granted under the First and Fourth Amendments? The answers to these questions, as well as those raised by Barlow on the nature of information and free speech, are important because they tell us whether our policies and practices serve us as well as they might. The issue is not simply hackers vs. system managers or law enforcers; it is a much larger question about values and practices in an information society.

Acknowledgments

I am deeply grateful to Peter Denning, Frank Drake, Nathan Estey, Katie Hafner, Brian Harvey, Steve Lipner, Teresa Lunt, Larry Martin, Gordon Meyer, Donn Parker, Morgan Schweers, Richard Stallman, and Alex for their comments on earlier versions of this paper and helpful discussions; to Richard Stallman for putting me in contact with hackers; John Draper, Geoff Goodfellow, Brian Reid, Eugene Spafford, Dave, Marcel, Mike, RGB, and the hackers for helpful discussions; and Richard Pethia for a summary of some of his experiences at CERT. The views expressed here, however, are my own and do not necessarily represent those of the people mentioned above or of Digital Equipment Corporation.

References

[1] Bruce J. Baird, Lindsay L. Baird Jr., and Ronald P. Ranauro. The moral cracker? *Computers and Security*, 6(6):471–478, Dec. 1987.

[2] John Barlow. Crime and puzzlement, June 1990. to appear in *Whole Earth Review*.

[3] Eric Corley. The hacking fever. In Pamela Kane, editor, *V.I.R.U.S. Protection*, pages 67–72. Bantam Books, 1989.

[4] Harry B. DeMaio. Information ethics, a practical approach. In *Proc. of the 12th National Computer Security Conference*, pages 630–633. National Institute of Standards and Technology, and National Computer Security Center, 1989.

[5] Peter J. Denning. Worldnet. *American Scientist*, 77(5), Sept.-Oct. 1989.

[6] Peter J. Denning. *Computers Under Attack.* ACM Press, 1990.

[7] Julian Dibbel. Cyber thrash. *SPIN*, 5(12), March 1990.

[8] Peter F. Drucker. *The New Realities.* Harper and Row, 1989.

[9] Lee Felsenstein. Real hackers don't rob banks. In *Positive Alternatives to Computer Misuse, A Report of the Proceedings of an ACM Panel on Hacking.* Assoc. Computing Machinery, 1986.

[10] Karen A. Frenkel. Brian Reid, a graphics tale of a hacker tracker. *Comm. ACM*, 30(10):820–823, Oct. 1987.

[11] Emmanuel Goldstein. Hackers in jail. *2600 Magazine*, 6(1), spring 1989.

[12] Emmanuel Goldstein. For your protection. *2600 Magazine*, 7(1), spring 1990.

[13] Geoffrey S. Goodfellow. Testimony before the subcommittee on transportation, aviation, and materials on the subject of telecommunications security and privacy, Sept. 26 1983.

[14] Katie Hafner. Morris code. *New Republic*, pages 15–16, Feb. 16 1990.

[15] Harper's. Is computer hacking a crime? *Harper's Magazine*, pages 45–47, Mar. 1990.

[16] Brian Harvey. Computer hacking and ethics. In *Positive Alternatives to Computer Misuse, A Report of the Proceedings of an ACM Panel on Hacking.* Assoc. Computing Machinery, 1986.

[17] Richard C. Hollinger and Lonn Lanza-Kaduce. The process of criminalization: The case of computer crime laws. *Criminology*, 26(1):101–126, 1988.

[18] Hans Huebner. Re: News from the KGB/wiley hackers. *RISKS Digest*, 8(37), 1989.

[19] Bill Landreth. *Out of the Inner Circle.* Tempus, 1989.

[20] John A. N. Lee, Gerald Segal, and Rosalie Stier. Positive alternatives: A report on an ACM panel on hacking. *Comm. ACM*, 29(4):297–299, April 1986. Full report available from Assoc. Computing Machinery, New York.

[21] Steven Levy. *Hackers.* Dell, 1984.

[22] Assoc. Computing Machinery. Just say no. *Comm. ACM*, 33(5):477, May 1990.

[23] John Markoff. Drive to counter computer crime aims at invaders. *The New York Times*, June 3 1990.

[24] John Markoff. Self-proclaimed 'hacker' sends message to critics. *The New York Times*, Mar. 19 1990.

[25] Larry Martin. Unethical 'computer' behavior: Who is responsible? In *Proc. of the 12th National Computer Security Conference*, pages 531–541. National Institute of Standards and Technology, and National Computer Security Center, 1989.

[26] Gordon Meyer. The social organization of the computer underground. Master's thesis, Northern Illinois Univ., DeKalb, IL, Aug. 1989.

[27] Gordon Meyer and Jim Thomas. The baudy world of the byte bandit: A postmodernist interpretation of the computer underground. Technical report, Northern Illinois Univ., DeKalb, IL, Mar. 1990.

[28] Tom Peters. *Thriving on Chaos.* Harper and Row, 1987.

[29] Pamela Samuelson. Information as property: Do Ruckelshaus and Carpenter signal a changing direction in intellectual property law? *Catholic University Law Review*, 38(2):365–400, Winter 1989.

[30] Eugene H. Spafford. The internet worm, crisis and aftermath. *Comm. ACM*, 32(6):678–687, June 1989.

[31] Richard Stallman. Letter to ACM forum. *Comm. ACM*, 27(1):8–9, Jan. 1984.

[32] Richard Stallman. Against user interface copyright. *Comm. ACM*, 1990. to appear.

[33] Guy L. Steele Jr., Donald R. Woods, Raphael A. Finkel, Mark R. Crispin, Richard M. Stallman, and Geoffrey S. Goodfellow. *The Hacker's Dictionary.* Harper and Row, 1983.

[34] Clifford Stoll. *The Cuckoo's Egg.* Doubleday, 1990.

[35] Jim Thomas. Review of The Cuckoo's Egg. *Computer Underground Digest*, 1(1.06), Apr. 27 1990.

[36] Jim Thomas and Gordon Meyer. Joe Mccarthy in a Leisure Suit: (witch)hunting for the computer undeground. Technical report, Northern Illinois Univ., DeKalb, IL, 1990. see also the *Computer Underground Digest*, Vol. 1, Issue 11, June 16, 1990.

A Reassessment of Computer Security Training Needs

Dennis F. Poindexter
Department of Defense Security Institute
C/O Defense General Supply Center
Richmond, VA 23297-5091

Abstract: For many organizations, training in computer security is a current and pressing problem: there is not enough time for it; there is not enough qualified instructors; the people who need it most avoid it; the content of alternative training methods is repetitive and too basic or too advanced for all but the extremes of students. We cannot expect basic improvements in the quality of training programs until we identify and correct the most pervasive problems of the discipline which produces and oversees the products being used to train employees. This requires both direction and management of the training programs of all participating agencies.

William Glasser in Schools Without Failure, says, "Children are dismayed by the sudden and to them incomprehensible difference between the first five years of their lives, when they used their brains for fun and for solving their own problems,...and their life later in school when, with increasing frequency from grade one through graduate school, much of what is required is either totally or partially irrelevant to the world around them as they see it."[1] It may be equally true of the computer business as a whole, but it is especially true of computer security training. The bulk of educators in the government training establishments do not have an Education background and are teaching subjects that are so diverse and technical that to be a subject matter expert is impossible. We must actually define what is trainable, what is educable, and most of all, what is necessary. Educators cannot do all the things that are now required of them.

In doing a limited survey of available courses for DoD, we found a considerable variety of subject areas being taught as computer security [2]. We divided these into two general categories, surety and security, examples of which follow:

Surety
Data Continuity
Content Integrity
Personnel Reliability
Continuity of Operations
Safety of Equipment
Fire Protection
Auditing

Security
Risk Analysis
Accreditation
Physical
Personnel
Magnetic Remanence
COMSEC/Transmission

All of these things are computer security in one form or another, but it is not possible to train them all in a reasonable amount of time. Those who try suffer a common malady: courses which were intended to be specific training to a performance standard, eventually became general overviews which are then not attended by their target audiences. The result is very little specific skills training for the very audiences that need it most. The basic problem is one of time. It takes considerably longer to train to a standard of performance than to make the same audience aware of an issue. This is the basic difference between training and education.

What is a "reasonable" amount of time to present material? From an education standpoint, it is relatively easy to arrive at an answer. One takes the subjects which must be presented, outlines the course, and factors in preparation and presentation times for each block of instruction, given the instructional method to be used. The result is the amount of time that it should take to present the material and evaluate the student's performance. The average number of hours of AIS security in the courses surveyed was 14.1. This means that in roughly two days of classroom time, the entire range of computer security issues can be adequately trained. Considering the stated objectives of some of these courses, it was difficult to believe that any trainer had set the number of hours at these levels.

We received unsolicited comments from several respondents relative to the amount of class hours given to the subject. All agreed that it was not enough, but was all they could offer. This frequent comment is a reflection of reality in the training world --- all the preparation of course outlines and relative hours needed are always subject to outside forces, particularly the unwillingness of many managers to part with their employees for much over a week at a time. In short, many of the difficulties of the training profession are the result of management decisions which fail to recognize the complexities of training to a standard. Given the restraint, there are three possible choices: (1) train less, but still meet the training goal for parts considered most critical to the performance; (2) do parts of the training through correspondence or video, in advance of the classroom training, or (3) redefine the objective to one of awareness. We found the latter to be more common.

Attempts to fit specialized training into short periods of time required students already well-rounded in the basics of computer security, yet few of the courses specified prerequisites, completion of advance materials, or actual experience in the field. Many directed their training by job function or grade of attendee, yet neither of these infer prior knowledge of the broad range of subjects required. Even where prerequisites were made, students frequently did not meet all of them and some came from disciplines other than the computer field. This tended to cause more class time to be spent on basic computer security issues. Ultimately, what specialized training that started at the inception of the courses, became overviews suitable only to a more general audience.

These generalized courses do not offer the type of performance training a specialist needs and the result is reflected in a survey of Air Force training in computer security. Even though adequate numbers of course offerings were made, nearly 1/2 of persons who hold the title of Information Systems Security Officer (or equivalent) had not attended courses designed and conducted for them [3].

While most agencies offer training courses, they tend to offer only one, as if it were possible to present all computer security areas at one time. A single course, offered under these circumstances, should be able to focus on agency - specific issues. To assure students are qualified for attendance, further restrictions must be made on who is allowed to attend and strict prerequisites established. Entering students must be roughly equivalent in their knowledge to attend specialized training.

This essentially means that introductory courses must be offered and supported across agencies. Two types of courses are required, those on computers and computer systems and those on security issues. The first is obvious; the second is not. Security is a discipline of which computer security is a small part. A basic understanding of Information, Personnel, Physical, and COMSEC Security, offered in most computer security text books, is essential to development of computer systems security concepts. One of our respondents said, with respect to these areas, "anyone can learn this stuff." It is true of almost any subject, but subject matter experts generally do the training and a background in computer systems rarely brings with it the type of expertise required. These must be prerequisites for more advanced training and more use must be made of subject-matter experts in these fields.

Second, agency-specific courses must be more advanced and directed towards tasks that must be performed, particularly accreditation of systems and auditing systems already accredited. These are not subjects easily addressed without a substantial knowledge base.

Absent prerequisite courses and improved audience management, we cannot hope to narrow and more accurately define specific course objectives. There is simply too much information to be covered in a reasonable time.

Last, agencies must identify resources to meet the training objectives and the courses to meet them. The stated policies of most agencies implied the availability of a great many courses, yet there were not many being offered. The information was difficult to develop because agencies charged with training responsibility frequently could not identify what computer security courses were offered for their activities. Potential students must find it exasperating to be unable to specify a training plan or budget for it, without the type of basic information required. The National Institute of Standards and Technology has undertaken to identify potential courses and maintain them in a database, but this can only be a first step towards developing common courses across agencies and specialized courses where they are needed.

To adequately budget, agencies must know the number of persons to receive training, course prerequisites and how they can be met, the minimum number to be trained annually, and that training which is mandatory for job performance. This aspect of planning requires some joint effort across agencies offering courses. Basic (introductory) courses should be standardized and established as prerequisites to more advanced training. In these, the focus should be limited. Lambert Gardiner in The Psychology of Teaching says "Some educators respond to the current criticisms about the educational system by stepping up the volume of information. Audio aids, visual aids, Band-Aids, teaching aides are all enlisted to increase the volume. It is like throwing water to a drowning person. Students do not need information as much as the skills for organizing it."[4]

Advanced training should be task specific, e.g. auditing of IBM systems , and open to those persons who perform the task, regardless of agency. Many of these courses may have to be contracted out because they require specific knowledge often not available in existing training centers. Career track training guides should be developed which outline mandatory and discretionary courses to be taken, based upon the position the person occupies.

Overall, training must be managed. People are of the most essential ingredient of a sound program. Their training is not something that just happens. Budgetary restraint demands cooperation from many different activities and realistic goals for the training establishment. We do not have them now.

References:

1. As quoted by Toffler, A., **Learning for Tomorrow**, Random House, 1974.

2. Department of Defense Security Institute, **"Report on Limited Survey of DOD Automated Information Systems Security Training"**, January 1989.

3. U.S. Air Force Center for Aerospace **Doctrine, Research and Educational Research Report No. AV- ARI 87-4, "An Analysis of the Air Force Computer Security Program"**, December 1987.

4. Gardiner, W. Lambert, **The Psychology of Teaching**, Brooks/Cole Publishing Company, 1980.

Information Security:
The Development of Training Modules

By
Corey D. Schou, Ph.D. and John Kilpatrick, Ph.D.
College of Business
Idaho State University

Information Resource Security Background

The Information Resource Manager is an individual responsible for all aspects of information processing from data entry to the Executive Information System.[1] He plays an important role in the security of the organization's information assets. It is critical that Information Resource Managers convey the importance of resource security to senior management of their organization. One possible source of the information resource security problem is illustrated by the following story.

Two hundred years ago, one patriot said, "I have but one lamp by which my feet are guided and that is the lamp of experience. I have no way of judging of the future but by the past." How then are we to avoid the pitfalls of the future both in our Information Society and as participants in the Information Revolution? These two terms describe extensions of the industrial revolution which has been molding the American Experiment for the last 200 years. We need a lamp to guide us through a few of the perils of the next segment of the path.

About one hundred fifty years ago Charles Dickens, in his novel *Hard Times* (1845), detailed the impact of some of the social and economic conditions being created by the industrial revolution. Dickens describes the dehumanization of the worker during this period of social upheaval and economic change. He shows the effects of management reacting to the workers as replaceable pieces or cogs of the physical system that they use to create their products. Management, not wise enough to be productive, was not able or willing view these cogs as individuals. This was the first flicker of light in the lamp of experience.

In the last fifty years, we have seen the beginnings of a new industrial revolution in which The Blue Collar worker initially welcomed job aids which increased his productivity and reduced his workload by adding a mechanical assist to his efforts. However, increasingly we see the typical Blue Collar worker replaced by an Iron Collar worker — represented by automated assembly processes or the robot. Since the Iron Collar worker neither bargains for wages, calls in sick nor needs an endowed retirement program; this Iron Collar worker improves the bottom line performance and therefore is judged to be good for business.

When the human problems associated with the introduction of the Iron Collar worker occur, we justify by referring to the relief from repetitive tasks and an attendant quality of life improvement for the workers. The change is ideal for those who are able to maintain their jobs or are able to be retrained; however, some individuals are not able to make the transition comfortably if at all. While I would by no means call a halt to this progress; I would remind managers to deal with the long range economic — human impact of their decisions on the long range bottom line.

Remembering *Hard Times* is important for the next generation because it seems to illuminate a new

[1] Portions of this appear in "Computer Security: Training Needs for Managers" `, *Data Security Management,* September, 1990.

series of socio-industrial changes. Just as Iron Collar workers spread throughout the industrial work place, the computer resource began spreading to the middle management class of the corporation. Currently the White Collar work force views the computer as a benign tool which supports his decision making process, reduces his workload and assists his intellectual efforts.

An examination of technological advances suggests that portions of the White Collar work force will be supplanted by the Silicon Collar worker - the computer and its associated support humans. The Silicon Collar worker is ideally suited for repetitive programmable decisions and actions with an attendant minimization of decisions at risk and uncertainty. The Silicon Collar worker needs neither stock options, Christmas bonuses, nor retirement plans. This Silicon Collar worker will supplement middle managers and improve their routine decision making more directly than any other change in this century. Initially the Silicon Collar worker is more susceptible to being undermined by modern Luddites and saboteurs.

In fact, it may be that we are beginning to see these modern Luddites already. Several years ago James L. McKenney in the Harvard Business review talked about the existence of an information archipelago in a series of articles. The author specifies three islands – data processing, telecommunications and word processing. The archipelago he describes and its attendant bridges is critical to most computing and information related activities in the modern business. The bridges (networks) are critical to the success of the modern Silicon - Collar Worker in that they carry the life blood - data[2].

An example of an attack on one of the bridges is explained by Clifford Stoll in his recent book, *The Cuckoo's Egg*.[3] He describes the deep personal anger he felt when he discovered that someone was prying at the door of his computer and violating his privacy. This modern Luddite was breaking down the fabric of the network - the threads of trust that hold an information society together.

As Information Resource Managers, we are responsible for assuring that the information resource is protected while it is available to all authorized users. These two roles are often antagonistic. To accomplish both goals, it is critical that we ensure that training include not only the traditional technical and operational subjects for which we provide training, material about ethics, security and privacy, as well as materials on social and economic issues. We should insure that these issues not be relegated to the list of materials presented when we have time, but should be woven into the corporate culture. Should we fail to do this weaving, the organization may enter the latest phase of the industrial revolution ill equipped to deal with the broader issues which will confront them.

Finally as we enter this new era, we should not forget that systems and networks are not made of printed circuit boards nor of wires nor of bits and pieces – they are made of people - individuals. Should we forget this we may encounter a new Hard Times for a modern day Dickens to write about. The following outlines some techniques for avoiding this problem through appropriate training of both line workers and management.

Security Role for Information Resource Managers

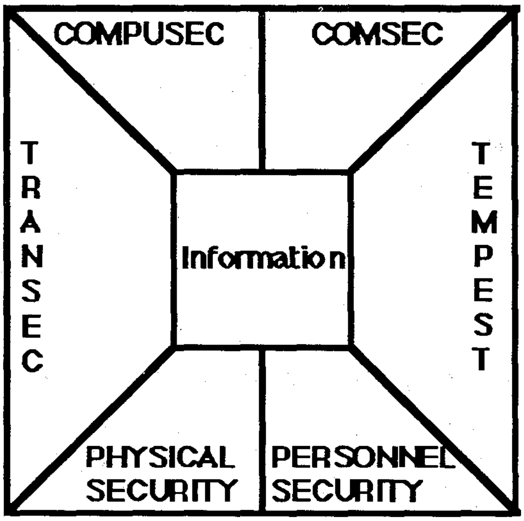
Information security is but one of the tasks of the Information Resource Manager. He is expected to provide everyone in the organization with his complete information needs while maintaining privacy and security of the information in their system. The manager must establish policies and pro-

[2] Schou, Corey D., "The More Things Change, The More They Stay the Same," *Scientific Computing and Automation,* February, 1990, pp. 5-6.

[3] Stoll, Clifford A., *The Cuckoo's Egg,* Doubleday, New York, NY, 1989.

cedure to carry out the latter half of his task. If well planned, they will establish a substantial framework for information security; however, these plans will be ineffective if the personnel involved are not trained adequately.

When developing an information security plan for the organization, it is important that the manager view it as an integral component of the overall information resource plan. One model for this was provided by Donald Latham (Assistant Secretary of Defense for Command Control, Communications and Intelligence) in his article *Security in the Information Age*.[4] He described an integrated model for information security. He pointed out that information is surrounded by a series of components which constitute information security. Latham's original model is shown in Figure 1.



Latham's Model
Figure 1

The Latham model as shown in Figure 1 surrounds information with six areas of primary concern:

TRANSEC
: Transmission Security - The protection of transmissions from interception and exploitation

COMPUSEC
: Computer Security - The protection of computer systems by electronic, software and procedural techniques

COMSEC
: Communications Security - The protection of communications by electronic techniques.

TEMPEST
: The means, usually through system design, to inhibit the unwanted emission of electromagnetic radiation from electronic systems such as cryptographic devices,

[4] Latham, Donald C., "Security in the Information Age," SIGNAL, May, 1987, pp.173-180.

telecommunications equipment, computers, etc.

PHYSICAL SECURITY

Physical measures to safeguard information

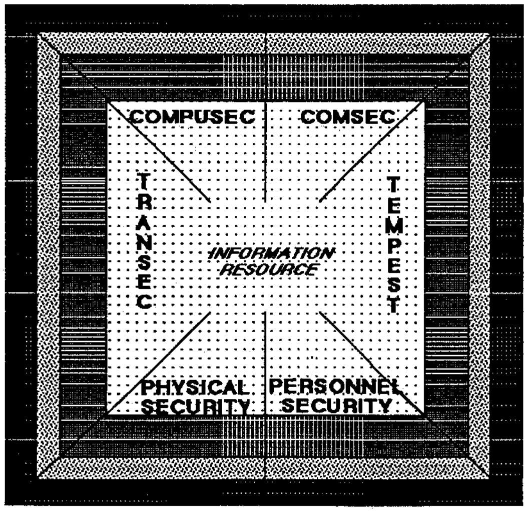
PERSONNEL SECURITY

Measures to ensure the integrity of people handling sensitive or classified information

Integrating Security into the Organization

In order to aid in integrating security into the organization, the Computer Information Systems faculty at Idaho State University, has developed a series of modules for use in the typical Information Systems curriculum. The initial problem we encountered in developing this course material was the determination of its overall contents. Originally it appeared that we could base our entire development on the U.S. Government publication *Computer Security Training Guidelines* by[5] Todd and Guitian; however, we decided to expand that basic approach. After using a modified Delphi technique, it became obvious that the Latham model could be extended.

Information security may be viewed as a series of shells shown in figure 2. The innermost of these shells represents the minimum skills for any knowledge worker in the modern world and is integral to the information itself. Each subsequent shell represents the skills required by individuals with increased responsibility for detailed information security. For example, the second shell is perhaps that which represents the skills of the Systems Security Officer (SSO), and the outer shell might represent the skills needed by the system evaluator.



Modified Latham Model
Figure 2

[5] Todd, Mary A., and Guitian, Constance, *Computer Security Training Guidelines,* NIST Special Publication 500-172, U. S. Department of Commerce (National Institute of Standards and Technology), November, 1989.

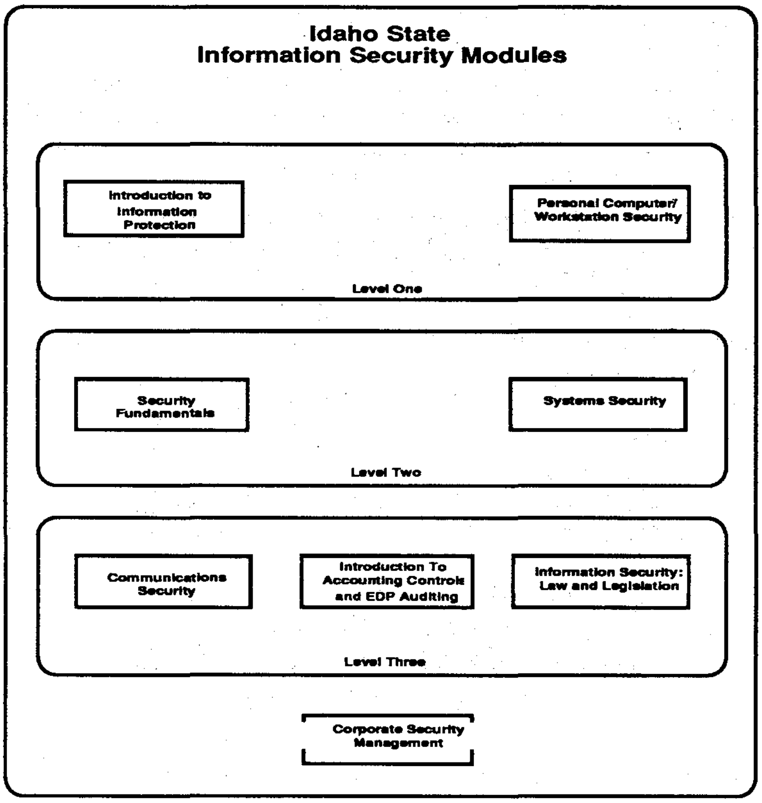
One should remember that beyond the innermost shell, the shells for any individual may not be symmetric. This lack of symmetry may be an indicator of increased training needs within the organization. To meet organizational training needs, the following modules are suggested.

Computer Security Training

These modules cover a spectrum of organizational needs

- Introduction to Information Protection
- PC/Workstation Security
- Security Fundamentals
- Information Security: Law and Legislation
- Systems Security
- Communications Security
- Corporate Security Management
- Introduction To Accounting Controls and EDP Auditing

With minor modifications, these modules can be applied as short courses outside the academic community. The relationship among these eight modules is shown in Figure 3.


Idaho State
Information Security Modules
Introduction to Information Protection
Personal Computer/ Workstation Security
Level One
Security Fundamentals
Systems Security
Level Two
Communications Security
Introduction To Accounting Controls and EDP Auditing
Information Security: Law and Legislation
Level Three
Corporate Security Management

Information Security Curriculum
Figure 3

The following details the contents of the modules. The Information Resource Manager can select from these to construct his own short courses within his organization.

Module one (Introduction to Information Protection)[6] is an introduction to information protection principles, and provides basic security concepts for all employees. Each major issue presented in this module is detailed further in its own module which can be used depending upon organizational needs.

I. Information as a Corporate Resource
II. Basic Information Systems Security Problems
III. Ethical Issues
IV. Areas of Information Systems Security Study

Module two (PC/Workstation Security)[7] introduces essential concepts of personal computer security and outlines components of an introductory training dealing with the basic security concepts of information processing. This module contains the fundamental knowledge for subsequent modules in this series. Topics include ethics and professionalism, security and data control, computer room environment, PC/workstation security familiarization, and physical security.

I. Ethical Use of the Computer
II. Computer Room Environment
III. Physical Security
IV. Data Security

Module three (Security Fundamentals)[8] is introductory in nature and provides basic security concepts for specialized Information Systems professionals. This module outlines and describes the basic requirements for planning, organizing and managing security in an organization. Topics include personal and organizational ethics; hardware and software issues; security and threats to data; recovery, control and audit procedures; and identification of corporate security costs and benefits. A bibliography and selected case studies are provided and may be used at the discretion of the instructor to expand any of these topics.

I. Planning
II. Organizational Policies & Procedures
III. Ethics and Professionalism
IV. Personnel Security
V. Physical Security
VI. System Security
VII. Threats and Vulnerability
VIII. Data Security and Recovery
IX. Control and Audit
X. Costs and Benefits

Module four (Information Security Law and Legislation)[9] may be used by middle managers who need to work with legal issues. It provides a legal framework for establishing company policies.

I. Underlying Problems
II. Laws as Tools for Computer Security
III. Laws and Legislation as Options to Control Computer Crime

[6] Watts, R.T., "Introduction to Information Protection," in *Information Security Modules*, National Computer Security Center, 1990.

[7] Burgess, J.D. & Watts, R.T., "PC/Workstation Security," in *Information Security Modules*, National Computer Security Center, 1990.

[8] Spiro, B., Schou, C.D. "Security Fundamentals," in *Information Security Modules*, National Computer Security Center, 1990.

[9] Richards, T., Schou, C.D. & Fites, P.E. "Information Systems Security Laws and Legislation," in *Information Security Modules*, National Computer Security Center, 1990.

Module five (System Security)[10] defines advanced requirements for security and the criteria on which satisfaction of those requirements can be judged. Hardware, software, firmware (software implemented in hardware), and procedures are considered as mechanisms to appropriately protect a system. The described process starts by defining the sensitivity of a system and proceeds through establishment of criteria and evaluation of the degree to which the criteria are met.

I. Overview
II. System Sensitivity
III. Security Requirements
IV. Levels of Security
V. Data Life Cycle
VI. Sample Protection Plan

Module Six (Communications security)[11] is intended to be taught to those who need to be familiar with data communications and networks. While several training perspectives could be used as the basis for this delivering module, this particular module is written from a management perspective for those who want to become intelligent users of such systems or those who aspire to a management role in an organization which relies on data communications systems or networks to interconnect elements of its information processing systems, either internally or externally.

I. Overview
II. Threats
III. Countermeasures
IV. Tradeoffs-Costs & Benefits

Module seven (Corporate Security Management)[12] deals with top management and policy considerations. Responsibilities of managers vary, depending on their level in an organization, and this module introduces differentiating responsibilities at various levels of management. One important role discussed in detail is the System Security Officer; he is responsible for all first line security within the organization. A corporate security management plan needs involvement by all levels of management to ensure that the program is properly and thoroughly implemented. The program should clearly identify local, state, and federal legislation which defines responsibility to ensure that all members of the corporation understand and are able to implement a specified plan. Ultimately, the corporation is held responsible for the accuracy and integrity of corporate data.

I. Overview
II. Development of Security Program
III. Risk Analysis
IV. Contingency Planning
V. Legal Issues for Managers
VI. System Validation & Verification (Accreditation)
VII. Information Systems Audit
VIII. Computer Security Check List

Module Eight (Introduction To Accounting Controls and EDP Auditing)[13] is an introduction to accounting controls and auditing concepts intended for a broad variety of students in courses where an appreciation of such concepts is needed. The module outlines key accounting and auditing concepts and describes the various roles played by management, information systems professionals,

[10] Walston, Claude, "System Security," in Information Security Modules, National Computer Security Center, 1990

[11] Walston, Claude, "Communications Security," in *Information Security Modules*, National Computer Security Center, 1990.

[12] Burgess, J.D., Schou, C.D., & Fites, P.E., "Corporate Security Management," in *Information Security Modules*, National Computer Security Center, 1990.

[13] Campbell, Terry L., "Introduction To Accounting Controls and EDP Auditing," in *Information Security Modules*, National Computer Security Center, 1990.

internal auditors, and external auditors. It deals with management control, application control, evidence gathering and evaluation, and management of the EDP audit function.

I. Overview
II. Roles
III. Systems Cycle
IV. General Internal Controls
V. Access Controls
VI. Input Controls
VII. Communication Controls
VIII. Processing Controls
IX. Database Controls
X. Output Controls
XI. Evidence
XII. Integration

These eight modules address many of the needs; however, the real need is for a discussion of the ethical environment.

Those involved in the design of a computer security system must be aware of the conflicting rights, responsibilities and needs of system users and professionals. By way of illustration, let's start with a number of paradoxical assertions:

- For people to have trust in a computer system, the systems manager must trust no one.
- Systems which are truly trustworthy must use control processes which inhibit use.

Another way of putting the problem, as Clifford Stoll suggests in his book, The Cuckoo's Egg, is that as "administrative controls" are added to ensure trustworthiness, the system becomes more difficult to use, which means that the people for whom the system is designed end up finding some other, less trustworthy but more easily accessible system to use. The term "administrative controls" refers to those organizational policies and procedures imposed by those higher in the organization and which are designed to regulate the individuals and activities covered by the policies and procedures.

Administrative controls are designed and implemented to make sure that people act in the way that organizational managers desire. Generally this means, “in ways that advance organizational objectives”. This may be something as simple as standardizing the ways employees claim reimbursements for job-related expenses. It may mean something as broad as the budget process, which attempts to regulate the activities of, and set standards for, entire departments, divisions or companies. Frequently, however, it also refers to the need to regulate behavior when it is perceived that a) there is motivation to engage in activities for personal, as opposed to organizational, reasons; and b) those activities are potentially harmful to the organization, to organizational values or to other organizational members.

If the interests of an individual always coincides with those of the organizations with which he or she lives and works, there would be very little need for administrative controls. It is at the point where these interests diverge that the need for controls arise. Further, some conflicts arise because of simple misunderstandings, some arise because of differences in perceptions, some are due to different priorities, world views or values, and some come about because of malevolent intentions on the part of some individuals.

Finally, there are those instances where it is an individual's self-interest for everyone else to exercise a degree of “moral restraint” while he or she exercises none. This can be seen as the “free-rider” problem or, to use Garrett Hardin's metaphor, it is the “tragedy of the commons”. In this environmental fable, the members of the community maintain their livestock on the commonly held grazing grounds. Animals can safely be added until the carrying capacity of the grounds are reached. However, it is to the benefit of any individual community member to add animals to his herd on the commons. The overall costs of degradation are borne by the community but the benefits accrue to the individual community member. The tragedy is that individuals can safely benefit in the short run while the long-term costs are dispersed. Greed is rewarded. The lesson is that, short of eliminating cooperative efforts, self-restraint is necessary.

It is beneficial to me if everyone else exercises discretion, judgment and professional respect for other's rights in the use of a computer information system. I know then that I can "trust" the system. It means that the system manager will be less concerned with intrusions or violations of rights and professional courtesies, respect and so on. But it also means that if I do desire to gain access to another user's files, to change data, steal information, study someone else's personnel file, install a Trojan horse or release a virus, it is much easier to do so. The implicit trust in the system makes it easy for me to violate that trust. Self-restraint can thus be seen as a prerequisite for any activity requiring trust.

The violation of the trust, if discovered, then necessitates a higher level of administrative control, new restrictions placed on access, additional procedural processes. My violations have caused a reduction in the efficiency and effectiveness of the system. A fundamental consideration, then, is to assess the role of trust, the desirable and achievable level of trust to be sought, and the implications of these choices for the organization and individuals affected.

This dilemma serves to highlight the ethical considerations facing the computer information professional. What balance between absolute confidence in the security of the system and completely free access for users is desirable? What are the tradeoffs between rights and responsibilities, costs and benefits implied by the security or control provisions that are contemplated? What values lie behind the choices made? As the level of security increases, and with it the consequent increase in the level of confidence or trust in the system, what other legitimate values are diminished or threatened? In general, this is the age-old question of the balance between individual and community interests. In specific terms, it is the question of how to optimize the legitimate and responsible use of computer information systems while eliminating unauthorized use and protecting the rights of users and other affected parties.

To generalize the problem:

- if people will not exercise "moral restraint", systems will develop controls for protection;
- the controls for protection will prove burdensome and inefficient;
- they will still be necessary as the threat comes, not from responsible users but from "mavericks" with what is arguably an essentially anti-community ethic.

Our training and education must also deal with values and choices. What is the role played by the "values" which members of a community hold and which inform the choices made? Stated another way, what is the significance of the way a member of the information systems community views the world and his or her relation to that immediate world and to other members of the community? These values and perceptions underlie the choices which individuals make, the goals that are pursued, priorities which are established. They affect both the means and the ends to which efforts are directed.

In a technological environment, it is easy to focus on the techniques designed to accomplish goals and on the technology used to assist in the accomplishment of those goals. At times the tendency is to allow the focus on "technique" to overshadow the purposes or ends. For example, a common observation of the modern "rat race" (a revealing metaphor) is that participants spend so much time and energy pursuing "the good life" that little is left over for living. In fact, a recent survey indicated that between 1973 and 1987, the amount of time available to adult members of our society had declined by ten hours. As this result suggests, it is easy to become obsessed with the tools and in the process to forget what the tools were designed for.

> Techniques are only means. To bring progress, there must be purpose; and purposes arise from values, which seem to be omitted from much of our formal education and which we are learning only with difficulty.
>
> [Behrman, 201]

> George Smiley: "You've made technique a way of life. Like a whore, technique replacing love."
>
> [LaCarre']

Our Information Security education should not fall into the trap of being relegated to an examination of techniques; we should encourage an individual passion for security.

DETERMINING YOUR TRAINING NEEDS

Adele Suchinsky
US General Accounting Office
441 G Street NW
Washington, DC 20548

Elizabeth Taylor is quoted as saying "I know the whereabouts and availability of every diamond of consequence on this planet."

We are all looking for the perfect diamond...that perfect training package that will do everything we want it to do, and our search can be exhaustive. But whether you are considering buying a diamond, a 4-bedroom house, or a training package, the criteria is going to depend on where you are coming from and what you want the product to do for you.

There is more information in the marketplace today than you need to know. You do not need to spend your time on the vast number of characteristics involved in evaluating all available options. Look only at the characteristics of a package as they relate to your training needs...DIRECTLY. I have seen Delphi studies for training with as many as 1,500 characteristics, but if you only need to be concerned with 12, then your primary concern is the 12 characteristics that you know you really need. What then, is your need? Will training solve the problem? How can you determine what is "right" for you?

NEEDS ANALYSIS is what you do before you do anything. It's the "front-end analysis," the "training needs analysis," and the "needs assessment." The Needs Analysis is all about asking the right questions and getting the right answers about your training program. It assists you in focusing on your objectives, approaches, strategies, and implementation.

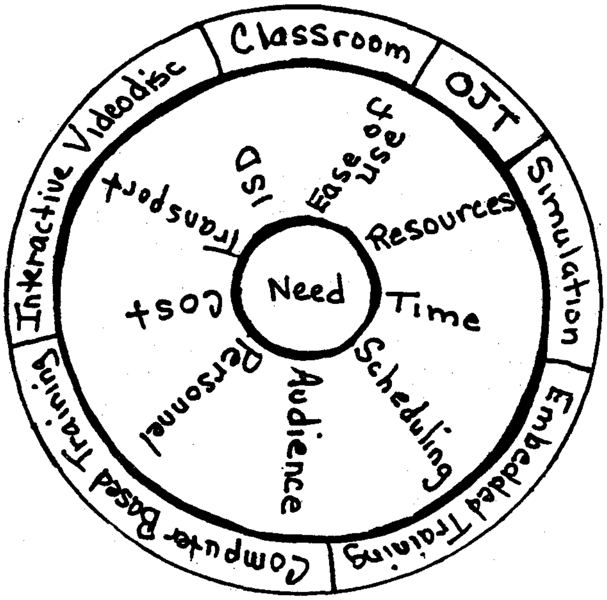
The beginning of any needs analysis is an identification of the training problem. The training needs analysis is an integral part of any well-designed training program. In order to utilize your training dollars efficiently, you must first determine the location, scope, and magnitude of the training need. Your primary purpose is to provide a competent work force by satisfying job-specific needs. Are you sure you have a "training" problem? Focus on the needs that will solve your problem. Recognize the difference between training needs and training wants. Determine what is required or expected in the job, and the degree to which this requirement is being met. This difference, or performance deficiency, identifies your training need. A Training Needs Analysis can determine that a job performance deficit is, in fact, a training problem, and that it can be corrected through a training solution. It is important to recognize at the outset that the "problem" is perceived from very different perspectives. For example, you have a new computerized format for completing the job at hand. You have also put the manuals for this new job aid on-line. Users are having trouble with the new tool. You have determined that you need "training" to teach users how to use this tool. But, is training what you actually need, or do you need better on-line instructions in the use of the program? Will a hand-held code list help? Perhaps a menu structure for accessing the on-line reference materials will work. Have users been given adequate time to adjust to the new format? Do they know how to use the computer? What are the supporting issues or tasks?

In your initial analysis, seek the answers to these key questions:

> Am I addressing an issue that affects how people perform?
>
> Is this performance critical and does it occur frequently?
>
> What do I understand about the current training, or the need for training?
>
> What do I observe about the needs which have been perceived?
>
> Can training actually make a difference or can this need be met through an alternative other than training?

Think about architects and builders. You decide to build a house, so you commission an architect to design it. The more the architect understands what you need and desire, the better he will be able to satisfy your needs. This understanding will also enable the architect to translate those needs into detailed blueprints, which the builder can use to implement those plans. This type of assessment becomes the foundation for your training program, as well. The better job you do in analyzing your needs, the easier it will be for you to develop your solution.

Picture a large circle. Inside, in the center of the circle, is your "need." Along the outside of the circle are your training options: Classroom training, On-The-Job Training, Computer-Based Training, Interactive Videodisc, Embedded Training, and the list goes on. But in between these two circles lie the factors which will influence your decision: Cost, Resources, Time, Scheduling, Audience, Personnel, Ease of Use, Transportability, Instructional Considerations, etc.



All these factors on all three levels overlap one another and affect the choices. It is here that you see the complexity of the problem every time you attempt to look at a training need. Consider, for example, one critical factor in any needs analysis -- the audience. Who is going to use this training? What training do I want to get across to this user? Who is the real audience -- management or worker? With classroom training, you can modify your training to fit the audience. With Computer-Based Training, this analysis must be determined up front.

Numerous models for conducting a needs analysis are available, and the alternatives for consideration vary greatly within each. All appear to share the identification of some goal or action based upon a performance shortfall. When conducting your assessment, consider the following:

What does the company want?
What do the users want?
What do the users know?
What are the users doing now?
What are the problems?
What is the level of experience of the user?
How does this translate into job performance?

Having determined a level of need, there are other factors affecting the decisions you need to make, as described in the example of the circles. Consider:

CONTENT (Subject matter)
Does it already exist in another format?
Are subject matter experts available?
Is it constant or changing?
How great is the need?
Is it applicable to the use of technology?
Does it require remedial branching?
simulation?
graphics?
video?

RESOURCES (Factors affecting implementation)

Is competent staff available?
Has a budget been established?
Is special equipment required?
What kind of space is needed?
Will contracting be required?
Are there contracting constraints?
How will the training be maintained?
How quickly do we need the training?

As stated earlier, don't consider ALL possible factors. Consider only those that are relevant to your training needs. The fact that your need may be to serve 2,000 students is, of course, relevant to your decision-making process. But, your selection of a format in which to deliver the training to 9 locations to reach these 2,000 students changes the parameters of the decision. Your first major step is to identify your key training needs. Once you have done so, you must set your priorities!

What do I need to have?
What would I like to have?
What compromises can I make?

This last factor is critical. You need to determine the highest level item on your list at which you can begin to compromise and make trade offs - in scope, in delivery time, in training format. You give up something to get something, but be careful not to "throw the baby out with the bath water." We reject a training solution too often because it does not do everything we want it to. We must seek alternative ways of including some of the things we want our training to do. If, for example, a particular package is available on the commercial market and has all the content you need, but does not test to the performance level you desire, can you do your own testing? If a package addresses the subject matter you need, but is not specific to your organization, can you add your own scenarios?

Picking the correct and appropriate problems and finding the best solutions are your keys to a successful training program. Needs analysis is a critical tool for doing just this. By learning how to ask the right questions, you can achieve a rational and logical assessment of the problems and potential solutions, and insure that your result will be useful and valuable.

Computer-Based Training: The Right Choice?

Althea M. Whieldon
Department of Defense
9800 Savage Road
Ft. George G. Meade, MD 20755-6000

I. What About CBT?

With the enactment of Public Law 100-235,"the Computer Security Act of 1987," all federal agencies must provide computer security training. Up until then, many did not have formal computer security training and now they are trying to establish such programs. Unfortunately, few off-the-shelf computer security training materials are available.

One way to deal with this training need might be to have the "subject," i.e., the computer, serve double duty as the training vehicle. This seems reasonable, as the "students" have access to computers. In practice, program managers in charge of setting up training must consider carefully whether computer-based training can best serve the training needs of their organizations. The program managers must first know what their training needs are. (A discussion outside the scope of this paper.) Next, program managers must examine what is involved in creating CBT, how it can be used effectively, and what its benefits and limitations are. Then they must decide whether CBT can fulfill their training requirements.

II. Methods of Creating Computer-Based Training (CBT)

Courseware can be created using almost any programming language. However, programmers must create their own special subroutines to handle specific training techniques, such as branching to different points in the program depending on the student's response.

As the demand for "authoring" products has grown, companies now offer educational products to assist in developing courseware. Authoring languages and authoring systems cater to the needs of course developers. Authoring languages are much like other high-level programming languages except the training subroutines are already built into the language. Like other languages, they require programming expertise to learn and use effectively. Authoring systems use menus, predefined screens and some embedded computer-aided instruction logic to allow non-programmers to create courseware. (7) These systems appeal to educators with little-or-no programming experience, because they prompt the developer throughout the creation of the courseware. Some authoring systems also include instructional management features to do things like register students, track students program and analyze lesson or test performance. However, they tend to impose the vendor's view of instruction design and can lack flexibility. For example, each authoring system has its own method of creating and linking screens for instruction. Some limit the number of possible student responses. If greater than the limit are needed, the developer may have to work around the problem and often the results are only marginally acceptable.

Developers of CBT should keep a perspective on the instructional need. Authoring features should be relevant to the training requirements, not just flashy. The products should be easy to learn and easy to adapt to the instructional design requirements. Among the most basic features an authoring software products should have are text editing, student interaction, branching, and testing capabilities. More advanced products could include graphics, sound, and external interfaces to other devices, like video.(6)

One other feature often overlooked is the quality of documentation and training for the product. Find out what type of documentation comes standard with the product and what kind is available at an additional cost. Also consider how accurate it is, how easy it is to understand, and how much time is required to learn how to use the product. (7) No matter how good a product is, if the supporting documentation is poor, the developers will have trouble using the product effectively.

III. Applications of CBT for Computer Security Training

Computers can boost the effectiveness of a training program if they are used appropriately. Too often, courseware is just an electronic page-turner. This type of design leaves adult learners frustrated and unmotivated. Even when enhanced with animation, courseware loses its appeal if it does not match the student's training needs.

Computers can be used for traditional training approaches as well as provide on-the-job self-help type applications. Several computer capabilities have educational application for computer security training.

Drill and Practice is a form of testing. (4) The computer presents a question to the student, and waits for a response. Once the student responds, the computer analyzes the input, provides feedback, and then branches to the next location based on the response. The intent of this technique is to reinforce prior instruction, not introduce new concepts. It can be designed to continue to provide material until the student chooses to end the session. It is patient and can be nonjudgemental. This form of instruction may have marginal applicability to computer security training. Although a drill and practice exercise might be useful to help users learn a new password.

Tutorial CBT has been dubbed the "electronic pageturner." (4) It presents information, elaborates on the concepts or procedures, and has quesitons or quizzes interspersed in the lesson. Well-designed tutorials provide students with challenging opportunities to interact with the subject matter. It is not just a mindless succession of hitting the enter key.

Simulations and games reproduce a model of an activity or process and are best suited for developing problem-solving skills. (4) While basically the same design, gaming includes a competitive element that often serves as a motivator to the student. This could offer some real challenges to computer security training, especially computer security officers. CSOs could track a phantom hacker through their "computer system" without inflicting harm to the real operations.

Socratic CBT establishes a natural language dialogue between student and computer. (4) Not only does the computer ask the student questions, the student is also able to pose questions to the computer. Stemming from research in artificial intelligence and expert systems, it requires considerable computing power to implement. This may be useful in building a case-study database of computer security incidents, threats, and vulnerabilities with associated countermeasures. As this form of instruction makes effective job-aids, this application could facilitate on-line queries of computer security policy and guidelines.

Embedded help provides the user with on-line information. (4) The user either enters a command or presses certain keys to invoke the help information. Information on various facets of computer security procedures could be available to all users of a system. For example, users could check the policy for maximum password lifetime. While this may make access to computer security policies more friendly for the users, it could also provide an adversary with additional insights as well.

IV. Pros and Cons of Using CBT

Using computers in education offers several benefits. CBT can provide remedial or supplemental instruction as needed to reinforce difficult concepts. Overall training time can be reduced. With the help of computer-managed instruction techniques, students can determine how to best focus their energies and not waste time covering material they already know. Training can be individualized for students, so they can determine the sequence of instruction and learn at their own pace. The computer can assess the student readiness for instruction based on the student's performance. Computers can alleviate some resource deficiencies. Through the use of CBT, instruction can be made more available at field locations or where there are teacher shortages. CBT can reduce expenses by replacing ostly lab facilities. High risk, rare , or dangerous procedures can be taught and practiced through computer simulations. (4)

CBT has its drawbacks. Producing CBT is time-consuming. Depending on the experience level of the CBT developer and the complexity of the training program, it can take from 100 to 1000 hours to develop one hour of instruction. Developing CBT is tedious and attention must be paid to every detail to make the courseware worth using. For disciplines that are undergoing constant revision, the maintenance costs associated with frequent updates of the material can be expensive. Another concern is having enough computer equipment available for the students or they will not get the training. In addition, if the training is to be available from the user's terminal, then courseware compatiability and portability become important. This is espcially critical in environments with a variety of computer equipments.

V. Decisions, Decisions

"CBT is an inherently active mode of learning. (4) To use computers as automated textbooks is an ineffective approach to training. Not only do adult learners develop a distain for CBT, but they can also lose interest in the subject matter in general.

To determine if CBT is the right choice, program managers must take the time to study the capabilities CBT systems offer. (Several excellent books on the subject are listed in the bibliography.) They must also identify the training needs of their organization and match those needs to the capabilities of CBT. In the end, to thy own needs be true. If CBT is the right choice, the organization's most critical training priorities will be satisfied.

References

1. Gery, Gloria, Making CBT Happen, Weingarten Publishers, Boston, MA 1987.

2. Hazen, Margret, "Instructional Software Design Principles," Educational Technology, November 1985, v25, #11, pp. 18-23.

3. Hord, Edwin V. "Guidelines for Designing Computer-Assisted Instruction," Instructional Innovator, January 1984, pp. 23-27.

4. Kearsley, Greg, Computer-Based Training: A Guide to Selection and Implementation, Addison-Wesley Publishing Co., Reading MA, 1983.

5. Kearsley, Greg, "33 Ways to Better Software Design," Training and Development Journal, July 1986, v40, #7, pp. 47-48.

6. Locatis, Craig and Carr, Victor, "Selecting Authoring Systems," Journal of Computer-based Instruction, Spring 1985, v12, #2, pp. 28-33.

7. Pollock, Joellyn, "Authoring Courseware: No Skills Necessary?", Educational Technology, November 1985, v25, #11, pp. 44-48.

8. Sadler, Rhonda R. and Coleman, Judi S., "CBT Evaluation: Why, What, Who and How," The 1987 Guide to Computer-based Training, Weingarten Publications, Inc., Boston, MA 1987.

9. Salisbury, David F. "How to Decide When and Where to Use Microcomputers for Instruction," Educational Technology, March 1984, v24, #3, pp. 22-24.

ANSSR: A Tool for Risk Analysis of Networked Systems

Deborah J. Bodeau, Frederick N. Chase, and Sharon G. Kass
The MITRE Corporation
Bedford, MA 01730

Abstract

DoDD 5200.28 requires that computer systems handling sensitive data undergo risk analysis to ensure that risks of information disclosure are known and, where possible, mitigated. ANSSR is a tool for performing risk analyses of networked systems, to aid in defining appropriate computer security requirements. ANSSR performs three levels of analysis, from a simple heuristic based on regulatory guidance to a detailed scenario-based analysis which shows how deliberate attacks could exploit system vulnerabilities.

INTRODUCTION

A risk analysis performed during the period when the functional requirements for a system are being defined helps to ensure that the computer security requirements are appropriate and adequate to counter the threats to which that system will be subject. For military computer systems, DoDD 5200.28 [1] requires such a risk assessment for information disclosure. Enclosure 4 of DoDD 5200.28 presents a high-level risk assessment procedure, in which a risk index is computed and mapped to an evaluation class from the Trusted Computer System Evaluation Criteria (TCSEC, DoD 5200.28-STD [2]) which defines a minimum set of recommended computer security requirements. Enclosure 4, derived from the Yellow Book (CSC-STD-003-85 [3]), provides a rough heuristic approach. It addresses networking concerns only to the extent that it extends the definition of "user" to include indirect users. It does not incorporate information about system attributes other than user clearance, data sensitivity, and development environment which could affect disclosure risk, such as user capabilities.

A number of extensions to the Yellow Book have been developed. Landwehr and Lubbes [4] extended the approach to address risk factors such as user capabilities. Logicon [5,6] further extended their work and developed a tool to implement this extended risk index algorithm. However, this extension does not address networking. Johnson and Layne [7] extended the Landwehr-Lubbes approach to address network connections, and developed a tool to implement their approach; however, their work does not include additional risk factors. One weakness of such heuristic approaches is that they do not show how computer security features counter specific threats.

The Analysis of Networked Systems Security Risks (ANSSR) prototype is intended for use during requirements definition. It includes three types of analysis: (1) a simple Yellow Book heuristic; (2) a more complex risk index heuristic, based on the Landwehr-Lubbes and Logicon work, extended to address risk factors associated with networking, including cascading; and (3) a scenario-based analysis, which shows ways deliberate attacks on a network could proceed and be countered by system security features. ANSSR allows the user to enter available information about the networked systems, whether more detailed (in the case of existing systems) or sketchy (in the case of systems whose concepts of operation and planned features are still being defined). For each type of analysis, ANSSR gives an overall network assessment, an assessment for each system, and the associated changes from the last time an assessment was run.

This work was sponsored by the U.S. Navy.

THE ANSSR PROTOTYPE

In the following subsections, we provide more detail on the intent, required information, and analysis techniques of the ANSSR prototype.

FOCUS ON COMPUTER SECURITY REQUIREMENTS DEFINITION

ANSSR is intended to aid computer security (COMPUSEC) requirements definition for networked systems, but to handle a stand-alone system as a degenerate case. It is also intended to be useful to network Designated Approving Authorities (DAAs), allowing them to identify weak points in networked systems and to consider the effects of architectural changes or additional safeguards. Because ANSSR is intended to aid in defining COMPUSEC requirements, its analysis is restricted to risks that can be mitigated by changes in the operational concept, the proposed network architecture, and the set of COMPUSEC safeguards provided by the systems in the network. In particular, it does not address risks due to the physical environment (e.g., earthquake, water main leak) or to physical threat mechanisms (e.g., bomb).

COMPUSEC requirements can include both preventative and mitigative safeguards. Analysis of the risk mitigation provided by recovery procedures requires detailed information not only about COMPUSEC features of the system, but also about operating procedures and organizational behavior. Such details are not available for planned systems and may be unavailable for existing ones. ANSSR therefore focuses on the effects of preventative safeguards (e.g., access control, user authentication) rather than those involved in detecting and recovering from attacks.

Disclosure risks due to natural disaster, structural failure, or human error are decreased by safeguards from disciplines other than COMPUSEC (e.g., physical security, quality assurance, human factors). Since ANSSR is intended to focus on COMPUSEC safeguards, it focuses on threats due to deliberate attack for assessing disclosure risk.

ANSSR does not provide a complete picture of the risks to networked systems. In particular, threats to availability and integrity are not addressed (except to the extent that disclosure could result from loss of availability or integrity). It is expected that trade-offs between different requirements will be made during requirements definition. ANSSR is intended to help in analyzing those trade-offs.

INPUTS

Three types of analysis are available to the ANSSR user: a quick application of the Yellow Book, an extension of the Yellow Book algorithm based on the work of Landwehr-Lubbes and Logicon, or a detailed analysis in which threat scenarios are developed. These analysis techniques are discussed below. The user can select a limited or normal level of detail of input; if the user selects "limited," menu selections related to physical and application interfaces will not be presented. (The "normal" option includes all "limited" input values as well as additional system and connection attributes.)

Figures 1 and 2 provide an overview of user inputs to the risk analysis, and indicate the types of analysis for which input values are needed. A graphic interface allows the user to draw the network topology as part of defining connections and interfaces.

Because ANSSR is intended for use when one or more networked systems do not yet exist, its analysis is based on high-level information about the networked systems. This information concerns the operational concept for each system, the proposed network architecture, and the set of safeguards provided (or proposed to be provided) by each system in the network. While detailed information on other safeguards (e.g., procedural, physical, TEMPEST) may exist for some of the systems under consideration, ANSSR does not use it for two reasons. First, the information may be closely held. Second, an imbalance of detail could skew the analysis.

ANSSR allows the user to define the set of sensitivity levels to be used. A sensitivity level consists of a hierarchical component (e.g., classification) and a category component; a category is understood to be a type of data for which formal authorization is required (e.g., Intelligence compartments, Special Access Programs). For each system, the user defines at least one disclosure asset, a collection of related information at a single sensitivity level. For the Extended Yellow Book and scenario assessments, the user also gives the volume, in number of facts, of each disclosure asset.

For each system, one or more user communities can be defined. A user is an individual who interacts with the system directly; if the system provides for identification and authentication, a user is someone who must identify themself to and be authenticated by the system. This includes both those individuals who use system devices (consoles, terminals) and those who use remote systems or dial-up lines to interact with the system. A user community is a collection of people with the same clearance (and formal authorizations, if applicable), the same capabilities, and the same type of connectivity to the system; that is, a user community is defined so that all members are interchangeable from the standpoint of threat scenarios. User capabilities range from invoking canned programs to system programming. Two special communities are assumed to exist: security administrators, who are capable of modifying authorization tables, and computer operators.

While ANSSR can analyze one or more stand-alone systems, it is intended to provide an assessment of networking risks. Three types of interfaces can be specified: physical interfaces, transport-level connections, and application-level connections. An interface is defined as one-way; to define two-way traffic, two interfaces must be defined to ANSSR. (This accommodates one-way hardware interfaces and connections in which traffic one way

Input	YB	EYB	SA	Values
● Disclosure asset attributes				
Sensitivity	x	x	x	(1)
Volume		x	x	in number of facts
● User community attributes				
Clearance/categories	x	x	x	(1)
Number		x	x	
Capabilities		x	x	query, full programming, ...
Local processing capability		x	x	dumb terminal, ...
Communication path		x	x	one- or two-way, speed, ...
● Maintainer attributes				
Clearance/categories			x	(1)
Number			x	
● Other system attributes				
Maintenance environment			x	open, closed
Need-to-know				yes/no (2)
Mode of operations				as defined in regulations (2)
TCSEC class				as defined in TCSEC (3)
IOC date	x	x	x	
End date	x	x	x	

(1) The set of possible classification levels and categories for data on the network is used to construct the menus for data entry.

(2) When all users are cleared for and authorized for all categories of data, either need-to-know or security mode of operations must be specified. Otherwise, these values can be inferred from the relation between user clearances/authorizations and data sensitivity.

(3) If a target TCSEC class has been defined for a system, that value can be input. A recommended TCSEC class will also be calculated based on the Yellow Book (for the Yellow Book and scenario analyses) or the Extended Yellow Book algorithm.

Figure 1. System Attributes

has different characteristics than traffic in the opposite direction.) Different threat scenarios involve exploitation of the different types of interfaces. For the Yellow Book assessment, only transport connections need be input. The existence of a transport connection from one system to another implies that some users of the destination system are indirect users of the source system. To accommodate assessments of an evolving network architecture, the user is prompted for the date of initial operational capability (IOC) and for the date when each system and connection will last be used.

Input	YB	EYB	SA	Values
• Transport connection				
Name				User-defined
Source system	x	x	x	already-defined system
Destination system	x	x	x	already-defined system
Minimum data sensitivity	x	x	x	(1)
Maximum data sensitivity	x	x	x	(1)
Average message size			x	in number of bytes
Maximum message rate			x	in number of bytes/second
IOC date	x	x	x	
End date	x	x	x	
• Physical interface				
Source system			x	already-defined system
Destination system			x	already-defined system
• Application interface				
Name				N
Source system			x	already-defined system
Destination system			x	already-defined system
Maximum data sensitivity			x	(1)
Application type			x	virtual terminal, job transfer, ...

(1) The set of possible classification levels and categories for data on the network is used to construct the menus for data entry.

Figure 2. Connection Attributes

ANALYSIS TECHNIQUE

Yellow Book Assessment

ANSSR implements the Yellow Book assessment technique on a system-by-system basis. For a given system, ANSSR examines the data assets defined for that system and determines the data sensitivity index (R_{max}). User clearances (and formal authorizations for categories) are examined (and, in the case of categories, compared with the sensitivities of data assets) to determine the minimum user clearance index (R_{min}). The risk index for the system is the difference between these two indices. If these are the same, either the need-to-know flag or the security mode of operations is used to determine whether a C2 system is appropriate. If the risk index is positive, it is used to determine the appropriate TCSEC class. If the user has input a target TCSEC class for a system, the computed class is compared to that target and the user is told whether the target class complies with the Enclosure 4 guidance. Otherwise, ANSSR simply informs the user of the recommended class.

Extended Yellow Book Assessment

The TCSEC class recommended as the result of the Yellow Book assessment is based on only three factors: data sensitivity, user clearance/authorization, and development environment. Landwehr and Lubbes extended the Yellow Book approach to include consideration of a wider variety of risk factors; the risk index is adjusted by the use of a number of tables. This table-driven approach explicitly depends on relationships among the risk factors.

Logicon extended the Landwehr-Lubbes approach to include more risk factors and more possible values of those risk factors. This approach is automated in the Effective Risk Management (ERM) Decision Support System (DSS); ERM can also be implemented manually. Because of the large number of variables in the analysis, ERM uses a weighted sum of numeric values associated with the values of risk factors as an adjustment to the Yellow Book risk index. This facilitates manual use of the ERM algorithm.

ANSSR builds on both these approaches. Where risk factors are clearly related, tables are used to adjust the risk index. Independent risk factors are assigned weights and numeric values and used to adjust the risk index produced by the table-driven assessment. A recommended TCSEC class is computed. As with the Yellow Book assessment, if the user has input a target TCSEC class for a system, the computed class is compared to that target and the user is told whether the target class complies with the recommendation. Otherwise, ANSSR informs the user of the recommended class.

For the network as a whole, ANSSR implements a test developed by J. K. Millen for cascading based on the concepts of sensitivity domains and path resistance. A domain is defined to be the set of data of a given sensitivity within a given system. A path of length one between two domains exists if (a) the domains are on the same system or (b) the domains have the same sensitivity and a direct connection at that sensitivity exists between the two systems on which the domains reside. A path of length n between two domains exists if there is a sequence of length-one paths between those domains.

The path resistance of a path of length one from domain A to domain B ($pr_1(A, B)$) is zero if (a) the domains are on the same system and the sensitivity of B dominates that of A (i.e., information flow from A to B is an upgrade) or (b) A and B are on different systems and there is a direct connection between them. If the domains are on the same system and the sensitivity of B does not dominate that of A, the path resistance is the risk index corresponding to the TCSEC class provided for that system (or, if no TCSEC class has been provided, the ANSSR-computed risk index). If there is no path of length one from A to B, the path resistance is infinite.

The path resistance of a path of length n (n>1) or less from domain A to domain B is defined as

$$pr_n(A, B) = \min_{C} (\max(pr_{n-1}(A, C), pr_1(C, B)))$$

The risk index associated with a path from A to B is computed as the Yellow Book risk index associated with a system containing data with A's sensitivity and having minimum user clearance/authorization for data with B's sensitivity. A cascading path from A to B exists if there is a path of length less than or equal to the number of domains in the network such that the path resistance is less than the risk index. If such a cascading path exists, ANSSR informs the user.

Scenario Analysis

ANSSR's scenario-based analysis technique allows recognition of (i) the interactions between different safeguards and (ii) logical and temporal dependencies among events in an attack. In addition, this type of analysis departs from the prescriptive/proscriptive ("Thou shalt"/"Thou shalt not") Yellow Book approach by offering instead a continuously-variable measure of residual risk implicit in operation of the network, as described. An ANSSR threat scenario is a succession of pre-identified events leading to an event with disclosure impact. Each scenario has an associated risk. ANSSR's scenario-based risk assessment process is (i) to identify every "different" scenario and associated risk to which the network is subject, and then (ii) to aggregate the risks in one or more meaningful ways. A scenario is initiated by a threat source; ANSSR threat source types are users (including security administrators and operators as special user communities), maintainers, customers, and outsiders.

Currently, scenarios are presumed to be orchestrated by a single individual, without accomplices. Future refinements of the prototype may include a recipient distinct from the original threat source and/or another innocent individual whose inadvertent cooperation is required.

Events

Individual events are the building blocks of scenarios. ANSSR recognizes a number of individual events, such as "threat source physically penetrates TCB perimeter," "threat source passes identification and authentication requirements as self (logs in)," and "threat source initiates an application connection to a port of a connected system." In addition to a verbal description, each event has an algorithm which, using as input the current state of the network (including attacker capabilities), (a) computes the likelihood (probability) that the event would succeed and (b) generates a new current state of the network. As an example of state alteration, a system's assurance factor is significantly reduced upon occurrence of the event whereby a user creates a flaw in that system's TCB.

It is possible to represent the decision of an individual to initiate an attack as an event ("user goes bad"). Such an event occurs early in every scenario. A single event with impact, and therefore a single event which can terminate a scenario, is currently identified. This event is the departure, by the recipient of the information in the impacted data asset, through the physical access perimeter for direct users of a specific network system.

Risk

The ANSSR scenario analysis allows either of two definitions of risk: single occurrence of loss (SOL) and annualized loss exposure (ALE). For a given scenario, *SOL risk* is defined as the probability of one or more impacts exceeding a threshold impact value during a chosen time interval, nominally a year. The threshold impact value can be altered by the risk analyst. Thus SOL risk for a given scenario is either zero (for a below-threshold impact), or the impact likelihood itself. For the same scenario, *ALE risk* is defined as impact value times impact likelihood.

These measures of risk were chosen, in part, so that risks can be aggregated. Aggregation of SOL risks is accomplished by combining independent probabilities. Aggregation of ALE risks is accomplished by adding the risks. In either case, risks being aggregated must be presumed to be independent. This is a strong assumption, and must be taken into account when the results are examined.

In either SOL or ALE risk, *impact value* is defined so as to be a function of (i) the data sensitivity, and (ii) the volume of information in the data asset. Thus, the impact of disclosure of a very small amount of Top Secret information could be exceeded by the loss of extensive amounts of Secret information. We express disclosure impact value solely in terms of the sensitivity and volume of the data disclosed (rather than as a function of sensitivity and recipient clearance), assuming that if an attacker has gone to the effort to cause data to be disclosed to a recipient, it is with the expectation that the recipient will misuse the data.

In either SOL or ALE risk, an *impact likelihood* is computed as the product of two factors. The first is the cumulative scenario success likelihood (CSSL), the probability that the entire sequence of events in some scenario can succeed, given that each is attempted in the context of the preceding events. The second factor is the *attempt likelihood*, an estimate of the psychological propensity of the individual to choose/attempt/initiate the particular subject scenario.

ANSSR estimates attempt likelihood by taking the product of three subfactors: (i) a *work factor* based on the number of events, (ii) an *apprehension factor*, and (iii) a *gain factor*. The gain factor is computed as cumulative scenario success likelihood times impact value, normalized with respect to the the maximum gain factor available to that threat source. Although the twofold use of CSSL in the calculation of risk may be questioned, its effect is simply to reduce the risk attributed to scenarios which appear to be less likely to be attempted. The second use of CSSL drops out for the most attractive scenario, since the normalized gain factor is 1.0 in that case.

The analysis begins with nested loops determining (i) threat source, (ii) threat source initial system, and (iii) user community (when the threat source is a user). A procedure then recursively makes subsequent eligible events "occur" until (i) an impact event occurs; (ii) the cumulative partial scenario likelihood, computed as the previous cumulative partial scenario success likelihood times the success likelihood of the current event, drops below a given threshold; or (iii) the scenario becomes too long. Each stack frame of the recursion has the cumulative partial scenario likelihood and a network surrogate as altered by all preceding events. To avoid a combinatorial explosion involving generation of improbable event sequences, each event has a pre-identified set of subsequent eligible events. This also prevents double-counting of scenarios which differ only in the order of mutually independent events.

As with every ANSSR analysis technique, a per-system risk is computed, and an assessment of network risk is made. While the Extended Yellow Book approach assesses network risk in terms of the existence of cascade paths, the scenario analysis approach expresses network risk as a combination of the individual system risks. These risks are quantitative and (essentially) continuously variable. Both on the graphic network view and in the textual report, the new risk value and the (percent) change from the previous value are presented.

As with any risk analysis technique, and especially with any technique that produces numerical results, the results of an ANSSR analysis must be caveated. There will be uncertainty about the correctness and accuracy of the algorithms for individual events. The algorithm for summarizing the results can introduce a bias either to underreport or to overreport risk. There will be uncertainty about the correctness and completeness of the system descriptions, especially since ANSSR is intended for use during the concept exploration phase. (For example, during that phase individual data assets such as files cannot be identified.) The factors, probability estimates, and risks computed by ANSSR carry more precision during intermediate computations than is meaningful in the final results. ANSSR therefore currently displays only three significant digits in reporting risk values, corresponding to an uncertainty of a few parts in a thousand. This is overly precise for risk assessment (but is useful for prototype validation); future work includes performing sensitivity analyses to determine how risks should be reported.

Scenario-based results are most useful for "what-if" exercises, in which different combinations of architectures, safeguards, and operational concepts are explored. The ANSSR scenario analysis can be regarded as an initial step towards quantification of disclosure risk. Meaningful risk quantification, together with cost analysis, has the potential to provide a more rational basis for design decisions.

IMPLEMENTATION

The prototype is implemented in the Smalltalk-80 environment, and currently runs on a Mac IIx. A minimal configuration appears to be a 68020- or iAPX286-class machine with ample memory and a hard disk.

CONCLUSION

There is a need for computer security risk management for mission-critical computer systems and for computer systems that handle sensitive data. The question of how to perform a risk analysis effectively is complicated by technological advances, which can create new vulnerabilities as well as provide new safeguards.

A variety of techniques for computer security risk analysis have been developed. Different techniques are appropriate under different circumstances. However, we can make several general observations. First, computer security risk management differs from general risk management in its special emphasis on disclosure risks and risks due to deliberate attack. The emphasis on disclosure means that a technique must provide a well-defined approach to estimating disclosure impacts to be useful. To address deliberate attack, it is useful for a technique to allow descriptions of how an attack progresses, which safeguards could prevent the attack from resulting in an impact, and which safeguards could facilitate recovery from the attack. Second, because the technological environment is still changing, it is insufficient to rely on simple heuristics. The assumptions on which those heuristics were based may have changed, and the heuristics generally are not accompanied by a description of those assumptions which would aid in determining whether they are still valid. Third, because computer systems are increasingly networked, a useful computer security risk analysis technique should address risks due to networking.

The ANSSR prototype provides a way of assessing disclosure risks to networked systems due to deliberate attack. It incorporates and extends the Yellow Book heuristic, to give the user a quick look at how well individual systems and a network as a whole comply with regulatory guidance. In addition, it provides a scenario-based analysis which identifies how attacks could occur, which attackers are most dangerous, and where specific vulnerabilities lie. Work continues on refining and extending the database of events, and on validating prototype results by analysis of actual networks. To determine which combinations could be acceptable to a DAA, future work could involve establishing a baseline by assessing the scenario-based risks to an already-approved network.

REFERENCES

1. "Security Requirements for Automated Information Systems (AISs)," DoDD 5200.28, Washington, DC: Department of Defense, 21 March 1988.

2. "Department of Defense Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, Washington, DC: Department of Defense, December 1985.

3. "Computer Security Requirements - Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments," CSC-STD-003-85, Fort Meade, MD: National Computer Security Center, 25 June 1985.

4. C.E. Landwehr and H.O. Lubbes, "Determining Security Requirements for Complex Systems with the Orange Book," *Proceedings of the 8th National Computer Security Conference*, October 1985, pp. 156-162.

5. S. Misgues, "A Guide to Effective Risk Management," *Proceedings of the Third Aerospace Computer Security Conference,* December 1987, pp. 66-86.

6. "A Guide to Effective Risk Management: Decision Support System," Arlington, VA: Logicon, Inc., 3 October 1988.

7. Howard L. Johnson and J. Daniel Layne, "Modeling Security Risks in Networks," in *Proceedings of the 11th National Computer Security Conference*, October 1988, pp. 59-64.

Approaches to Building Trusted Applications

Helena B. Winkler-Parenty

Sybase, Inc.
6475 Christie Avenue
Emeryville, CA 94608

Abstract

Trusted applications are needed to provide powerful tools for users of trusted systems. Two common problems, one in the commercial sector and one in the military arena, are presented; both of which require a Database Management System (DBMS) as part of their solution. This paper focuses on trusted DBMS's to explore the issues surrounding trusted applications. Security architectures and evaluation strategies of three DBMSs are described and contrasted with each other. The Integrity Lock approach has known security vulnerabilities that make an assurance level of B2 or greater difficult or even impossible to achieve. Hierarchical TCB subsets constrains application designers, potentially forcing inefficiency, but the evaluation cycle is reduced. TCB subsets augmented with trusted subjects allows the application designer greater flexibility and control, but potentially requires a longer evaluation cycle.

Motivation

The majority of the end users of computer systems interact with an application and not directly with an operating system. This is because it is not cost effective for each user to write, debug, and maintain their own set of programs. Instead, they use either off the shelf or special purpose software developed for a particular customer's needs. Accounting packages, spread sheets, editors, and database management systems (DBMSs) are just a few of the many examples of applications.

Users of trusted systems, like users of non-secure systems, are considerably more likely to have an application, rather than an operating system, as their primary interface to a computer system. This means that the security implications of both the operating system and the application need to be considered. In the security arena a trusted system must meet both the security and computing needs of the customer. If the security requirements aren't met, the system cannot by deployed. If the computing requirements aren't met, the system will not be used.

This paper explores architectures and evaluation strategies for trusted applications. Security, ease of development, portability, extensibility, and performance are analyzed for each approach. The latest draft of the Trusted Database Management System Interpretation (TDI) [1] is used to compare evaluation approaches. The TDI is an interpretation of Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) [2] for a specific class of application, database management systems. It provides a general taxonomy of security architectures, and guidance on evaluation considerations.

Application Examples

Two archetypical examples, one in the military world and one in the commercial world, are presented to discuss the capabilities needed by a trusted application. Every government needs to address the following problem. In times of crisis, the chiefs of the armed forces must be able to quickly obtain information on the location, number, and type of all personnel and equipment. Typically, this information will be stored in a database. The database can also used to store military orders, and the progress being made in carrying out each of the orders. The security levels of the military information and personal clearances are determined by the government. In many cases, the clearances of military personnel will rise as they rise in rank, and compartments are used to separate information on a need to know basis. A colonel would typically be cleared to a higher level than a lieutenant colonel, and the Army Chief of Staff needs to know different information than the Navy Chief of Staff.

Similar to the example of a government managing military information, commercial businesses manage information on their operations, including personnel, assests, expenditures, and other company related information. One instance of this is a budget planning process.

In a large corporation first line managers calculate their groups' budget, and pass this information onto their managers. This process continues iteratively up the management chain until the vice presidents of each department have preliminary budgets. The vice presidents and president of the corporation determine the company's final budget, which is then passed down the management chain.

Confidentiality is critical during a budget planning process, since salary information is part of a department's budget. Mandatory access controls can easily be used to protect budget information. Each department is assigned a unique compartment, and each level of management is assigned a hierarchical level. Higher levels of management correspond to increasingly greater security levels. This division allows a second level manager to view the budgets of all first line managers reporting to him/her, but not the budgets of managers either in other departments or above him/her.

There are many other examples of applications containing sensitive data. A computerized record keeping system for hospitals stores and must protect the medical records of its patients. A computer application controlling a chemical processing plant contains the company's most vital information - the formulas and processes which produce the company's products. These two applications, as well as the first budget planning application, typically contain a DBMS.

DBMSs are both an application and part of the underlying system upon which an application can be built. Increasingly greater amounts of data are being kept in computers, and DBMSs are frequently utilized to provide efficient data storage and retrieval mechanisms, data consistency and correctness, transaction control and recovery facilities.

Throughout the rest of this paper a DBMS will be used as an example to better understand the challenges associated with building trusted applications. A DBMS was chosen both because a DBMS is a common application, and because DBMSs are frequently embedded in other applications.

Application developers create new systems by manipulating the primitive objects and functionality provided by the base system. The new user interface created, and its primitives, can vary greatly from the underlying system's interface. For example, the UNIX shell is the user interface to a UNIX operating system, and the primitive objects are files and directories. SQL is the interface to many relational DBMSs, and the basic objects are rows, tables and directories. It is important that the base system, whether it is an operating system or an application, provides a wide range of powerful and flexible mechanisms and objects. If so, a large number of diversified applications can be built to extend the functionality of the system.

Previous Work

Overview

The approachs promulgated since the 1970's have placed most, if not all, of the security features and assurances in a secure operating system kernel. In contrast, the outer layers of the system contain few, if any, security features or assurances. This approach can be applied directly to operating systems or applications. The motivation has been to build applications that can take advantage of the operating system kernel protection mechanisms that have already been built, evaluated and tested, thereby saving development time and cost, and reducing evaluation risk.

Examples of systems based on this model are presented in this paper. These are: KSOS, an operating system; Trudata, a multi-level secure (MLS) environment for a DBMS; and Seaview's research project, an MLS DBMS. For each system its architecture is described. Information on inherent security vulnerabilities, guidance from the TDI on evaluation considerations, and general comments on ease of development are included. The paper will further analyze each of these projects with respect to extensibility, portability, state of the art technology, and performance. An alternate model, placing more security features and assurances in the application, is presented using the SYBASE Secure SQL Server™ as an example.

KSOS is an operating system comprised of both trusted software, and an untrusted UNIX emulator which provides a UNIX interface. The security kernel mediates every access attempt by each user process, and enforces both mandatory and discretionary access controls.

Trudata, a trusted DBMS environment implementing the integrity lock architecture, contains a trusted filter, a trusted operating system, and an untrusted DBMS. The filter mediates all access between users and the DBMS, and performs a trusted downgrade function when providing users at lower security levels with data from the database. A trusted operating system, at the B1 level or higher, is needed to provide mandatory access control for all users and user processes. In addition to the operating system mandatory and discretionary access control mediation (MAC and DAC), the security filter enforces the MAC and DAC policy of the DBMS.

The Seaview project has designed a MLS DBMS based on hierarchical TCB subsets architecture. The majority of the assurance and security functionality is located in the underlying operating system; the DBMS is only required to provide DAC and the other features and assurances found at the C2 level of trust. The operating system mediates all access to objects based upon mandatory and discretionary access control, the DBMS provides additional discretionary access control.

KSOS, A TCB Augmented With Trusted Subjects

KSOS, Kernelized Secure Operating System, was designed to be a trusted operating system [5]. KSOS is composed of three components: the security kernel, the UNIX emulator, and the non-kernel system software. The security kernel's function is to provide a simple operating system that can be demonstrated secure. Through a combination of both hardware and software checking, it mediates every access attempt by each user process, and only permits those accesses which comply with the access control policy of the security kernel. The UNIX emulator provides a UNIX-like user interface. It transforms the user's UNIX operating system calls into calls to the security kernel. The non-kernel system software is a collection of trusted processes performing support services. These processes have the privilege to selectively violate the rules, specifically the *-property, of the kernel.

The security kernel creates the objects exported by its interface from the base hardware resources, and enforces MAC and DAC on these objects. Objects provided by the security kernel include processes, files, and devices. Since the UNIX emulator was only one of many possible security kernel applications, few UNIX specific structures were placed in the security kernel. For example, the security kernel has no knowledge of UNIX directories, only the UNIX emulator contains this information. The UNIX emulator creates UNIX-level objects from kernel-level objects. As examples of this, it creates the UNIX file system from the simpler security kernel file system, and it caches block I/O to provide UNIX byte I/O.

System maintenance and administration services are provided by the non-kernel system software. Many of these operations need to violate the security policy enforced by the security kernel, in order to operate properly. These services are needed to meet the needs of a real user community.

Although the TDI is being written to assist with DBMS evaluations, it covers system architectures and evaluation strategies which can be applied to applications other than DBMSs. Therefore, its observations are relevant to KSOS, an operating system.

KSOS contains both a security kernel and trusted subjects. The TDI states that when a TCB is augmented with trusted subjects, both designers and evaluators of the trusted subjects must understand the base TCB. In addition, some of the base TCB may need to be examined during evaluation of the trusted subjects. Neither of these points are of concern for KSOS, since the trusted subjects were designed as part of the initial system.

There are no known security vulnerabilities inherent in this architecture. The TDI highlights three areas for scrutiny: relative privilege, resource protections, and covert channels. KSOS must provide appropriate assurances that the trusted subjects do not violate the portions of the security policy assigned to them, and that any objects implemented by a trusted subject can only be accessed when using the protection mechanisms of that trusted subject. The covert channel analysis of each trusted subject may involve covert channel analysis of the entire TCB.

A system composed of a security kernel and trusted subjects lends itself to a composition of components by more than one vendor. The TDI draws attention to potential vulnerabilities that may arise when several teams, working independently, build a single TCB. These concerns are not relevant for KSOS, since the same organization designed and built the entire TCB. SCOMP, an operating system evaluated at the A1 level of trust, is an example of an operating system comprised of a security kernel and trusted subjects, that was built by one organization.

The effort required to build KSOS was made considerably easier by the inclusion of trusted subjects. The designers of KSOS had wanted to build a small, provably secure, security kernel. In the process of achieving this goal they, as many other system designers have discovered, realized that their security kernel was too idealized for a system that would be used in an everyday environment. It is interesting to note that the mapping of UNIX-level objects onto kernel-level objects proved easier to achieve than the mapping of real world requirements onto the security kernel.

Trudata, Integrity Lock Architecture

Architecture

Trudata provides a multi-level environment for DBMS processing [3]. A trusted filter mediates all communication between the untrusted DBMS and the users of the DBMS. The trusted filter is responsible for labeling input data and restricting the data that the user can access, based upon the user's security clearance.

All primary data objects (e.g. tuples, elements) in the database have a security label associated with them. The trusted filter preprocesses all data input to the database by determining the data's security level and computing an unforgeable authenticator (i.e. a cryptographic checksum) over the data and its associated security level. This triplet, data object, security level, and checksum, are all stored in the database. Alternatively, the checksum could be stored in the filter.

The database is contained in a storage object classified at the highest level of data that can be contained in the database. The DBMS process runs at the same level as the database, and each of the user processes operate at the user's security level. The MLS environment is provided by a B1 or above operating system. Without this, functions such as the correct labeling of data input to the DBMS could not be accomplished in a trusted manner. Trudata relies on System V/MLS, a B1 operating system, and IDM, a ShareBase database machine.

Additional untrusted DBMS code runs at the user's process level for querying the database. Data retrieved from the database is passed to the filter, which confirms the security level associated with the data by recomputing the checksum. Assuming that the checksums match, the security filter will then determine if the user is cleared to view the data. If the mandatory and discretionary access control checks are passed, the data object is transferred to the user's process. If the checksum test fails, then the untrusted DBMS has permitted an incorrect security level to be associated with the data object, and an error condition will be raised. A key function of the filter is to downgrade data from the level of the database to the user's security level.

Vulnerabilities

One vulnerability with the integrity lock approach is the possibility of the mismanagement of the data, data authenticators and encryption keys. This threat can be reduced by storing the data authenticators (i.e. cryptographic checksums) in either a separate relation, or even within the filter itself. While storing the data authenticators and data separately improves upon the security of the system, it decreases the performance. In addition to retrieving the data pages from the database, a separate set of pages containing the data authenticators will also have to be retrieved and scanned.

A major vulnerability with this approach is the potential for the following covert channel. The untrusted system high DBMS has access to the entire database. Therefore, it can release highly classified data to lower level users by retrieving lower level data identical to higher level data. The filter will allow this low level data to be returned to the user's process, not realizing the true significance of the data. This threat is an inherent vulnerability of the integrity lock architecture.

Another serious problem is that the DBMS can return data which doesn't correspond to the query submitted. For example, the database is queried for a list of enemy locations to attack, but the data returned corresponds instead to friendly troops. There is no way to completely eliminate this risk. While it is unlikely that off the shelf DBMSs will contain such trojan horses, there is no guarantee that they will not. Nor is there any guarantee that trojan horses will not be introduced during the DBMS's life cycle.

Evaluation and Development Cost

The integrity lock architecture is a special case of the trusted subject approach, utilized by KSOS. In addition to the points raised in evaluating trusted subjects, the TDI requires that the evaluator determine that the following points are true: the portion of the DBMS used for querying the database should not be able to directly access the database; all data retrieved from the database must be mediated by the trusted filter; the trusted filter must place security labels and cryptographic checksums on all data records before they are passed to the DBMS; on retrieval, the trusted filter must validate security labels by recalculating the checksum, and only release data if the security level of the user's process dominates the level of the data; the operating system's TCB is B1 or higher; and no untrusted code can circumvent the policy enforced by the filter.

An attractive aspect of the integrity lock approach is the ease with which it can be implemented. Trudata was able to take two commercial products, and in a relatively short time develop a trusted filter. Historically, this approach has been viewed as a short term, stop-gap measure. It is viewed as short term because it is easy to implement, but it is only a stop-gap measure because of the numerous security vulnerabilities outlined above.

Better security can be obtained by storing the cryptographic checksums in the filter, but this will not substantially improve the security of the system. In addition, it would require the inclusion of DBMS functionality in the filter to store and retrieve the checksums.

Seaview, Hierarchical TCB Subsets

The Seaview research project has designed a trusted DBMS based upon the hierarchical TCB subsets architecture [4]. This approach decomposes the TCB into a set of hierarchical subsets, from a most privileged subset to a least privileged subset. The separation of subsets within the TCB is based on the allocation of parts of the overall security policy to each of the TCB subsets. Less privileged subsets may not violate the security policy of more privileged subsets. In particular, subjects in a less privileged TCB-subset cannot violate the policy enforced by the more privileged TCB-subset. This differs from the use of trusted subjects in KSOS and Trudata.

The most privileged subset in Seaview is GEMSOS, a security kernel targeted at the A1 level of trust. This subset enforces the mandatory and discretionary access control policies of the operating system. A less privileged subset contains Oracle's DBMS, which is responsible for enforcing its own DAC policy. The DBMS does not have a MAC enforcing role.

GEMSOS labels segments, requiring Seaview to place DBMS objects, namely elements and rows, into segments that the security kernel has labeled at the same level.

Seaview is based on a security architecture that has received extensive community review. Every subset is tamperproof, the most privileged subset enforces MAC, and no subset can violate the security policy of more privileged subsets. The extent to which previously evaluated subsets may need to be reevaluated during evaluation of the new composite TCB is at the discretion of the evaluator. One intended benefit of this architecture is the minimization of such reevaluation.

Modifying Oracle to conform to the constraints of the MAC policy of GEMSOS is a significant engineering challenge. Transaction control and recovery, complicated functionality in every DBMS, can be especially challenging in this environment. The effort saved by not including mandatory access control in the DBMS, may be negligible in comparison to the work involved with writing or rewriting access methods to retrieve data from multiple files, based on the user's security level.

Additional Computing Needs

There are many considerations involved when choosing a computer system. The foremost consideration for trusted systems is the security features and assurances of the system. Three trusted systems, each based on a different security architecture, were presented in the previous section. In addition to security, performance, state of the art technology, portability or connectivity, and extensibility are important attributes.

Performance is a necessary component in any system. Satisfying customer demand for high performance computer systems is always a challenge. This challenge is exacerbated in the security arena by the additional processing needed to satisfy security requirements. There are too many secure systems that are called secure as a brick, because they are too slow to be useful.

Nobody wants to choose between modern technology and security. State of the art technology is necessary to provide customers with the same functionality that non-secure systems contain.

Users typically have a preferred computing environment in which a new application must operate. Utilizing a client/server architecture, users only interact with the client side, requiring only the client application to be portable to a number of different platforms. The server can run on a smaller number of platforms, limited only by the client/server connectivity requirements.

Computer systems must be extensible. Few people work directly with an operating system, unless they are building an application for others to use. Most DBMSs, applications themselves, are embedded in other applications. Accounting packages contain spread sheets which are based on DBMSs which are built on operating systems. The UNIX emulator in KSOS can be viewed by the operating system kernel as an application. Computer software must have a user interface that allows for future growth and expansion.

KSOS

KSOS was designed to be extensible. To allow for many other applications few UNIX specific structures were part of the security kernel. UNIX, a popular operating system, was built on top of the security kernel to give users a modern operating system interface. One of KSOS's design goals was performance comparable to other UNIX systems. Since each UNIX command is run as a separate process, process generation needed to be a cheap operation. Unfortunately, the execution of UNIX FORK and EXEC were exceedingly slow, causing KSOS to be benchmarked as more than 10 times slower than its UNIX counterpart. The fundamental problem was that the interface and resources presented to users by the UNIX emulator differed too much from the interface and resources provided by the security kernel. KSOS was not a success due to lack of performance.

Trudata

The Trudata product contains several of the additional computing needs required by the user community. Its main disadvantages are that it does not contain a trusted DBMS, and there are serious security vulnerabilities with this approach. Nevertheless, because Trudata is based on several commercial products, specifically AT&T's B1 System V/MLS and ShareBase's IDM, it provides users with state of the art technology.

DBMS applications built on Trudata must conform to the fundamentals of the Trudata data model. Views are the level of granularity chosen for data labeling and protection. Therefore, data in the DBMS is accessible only through views. Baseviews are created automatically for each table, they exactly match the underlying table definition.

The trusted filter parses all input from users. It was initially constructed to parse ShareBase's version of SQL. Porting the filter to another trusted operating system would probably be a simple task. Modifying the filter to parse other SQL versions is a more time consuming project.

Early benchmarks figures indicate time estimates of approximately 120 milliseconds for command filtering and 130 milliseconds per object retrieved for data filtering. INTERCON Systems considers this an acceptable amount of performance degradation.

Seaview

Similar to Trudata, the Seaview Research project contains commercial operating system and DBMS products, namely GEMSOS and Oracle, thereby providing users with modern technology. Unlike ShareBase's DBMS which is partially implemented in hardware and is therefore completely unportable, Oracle's trusted DBMS is portable because it is written in software, on top of an operating system, independent of the underlying base hardware. New applications can be built on top of Seaview, within the constraints of the security policy enforced by more privileged TCB-subsets.

Seaview relies upon the operating system to define and implement the mandatory access control policy of the entire system. This may affect performance, since the operating system may not have implemented an ideal mandatory access control policy for all applications which may run on the operating system. In the field of non-secure DBMS architecture, Michael Stonebraker [6] showed that operating systems often provide the wrong services for DBMSs, and that severe performance problems are often inherent in using the operating system provided services. This problem is a consequence of the fact that the objects provided by the operating system are not appropriate objects for the DBMS. The same problem arises when the DBMS must make use of the protection mechanisms of the trusted operating system [7].

In Seaview the DBMS security object is an element or a row, whereas the GEMSOS security object is a segment. Since the DBMS is forced to divide the contents of a table, a logical unit of access, into a number of segments, performance degradations may occur. If degradation occurs, it will be dependent on the data distribution and the number of different security levels in the table. This degradation results from the operating system security policy determining how the DBMS manages its resources. Relying upon the protection mechanisms of the underlying operating system does not address the inherent architectural limitations of fitting an application as complex as a DBMS into the framework provided by an operating system.

For each level of data being read or written a separate segment has to be accessed and scanned. If only ten security levels satisfy the user's query, at least ten data pages will need to be read. In a database that doesn't rely upon the operating system to provide MAC, as little as one page may need to be accessed to retrieve the data for the user.

The example above assumed that there were no indices on the data. If no index is used to access the data, all of the data in the table(s) being referenced by the user's query may need to be read. Depending upon the distribution of data based on security levels, the number of pages access may be similar to that of non-secure DBMSs, or may be considerably higher. The larger the number of additional pages needing to be accessed the slower Seaview will perform.

Typically clustered and non-clustered indices are created on tables to decrease the number of pages needing to be retrieved. Once a clustered index is created, data with similar keys are physically located next to each other. An index allows data to be retrieved quickly, without necessarily needing to scan the entire table. An additional benefit of a clustered index is realized by queries which access data with the same or similar keys, because the data will reside on a small number of pages. Because of the database structure imposed by GEMSOS, only data with similar keys and the same security level can be located together on the same page. In addition, a separate index must be created and maintained for each security level contained within the table, resulting in considerable overhead as the number of security levels increases.

For example, consider a table with 10 rows that is small enough for both the index and data to fit on one page. (This simplified example is being used for explanatory purposes. DBMSs typically contain tables that are orders of magnitude larger.) This is a best case scenario, and only one page will need to be retrieved to respond to a user's query. With Seaview separate pages need to be accessed for each security level. For 10 rows of data at 10 different security levels Seaview will need to access 9 additional pages over a non-secure DBMS. Data retrieval and modification is a fundamental part of a DBMS. If it cannot be performed efficiently the overall performance of the DBMS will suffer.

Trusted Subjects

Three different projects and approaches toward building trusted products have been considered. The last approach considered contained possible performance penalties inherent to hierarchical TCB subsets, the architectural approach chosen. The previous approach contained security vulnerabilities, inherent to the integrity lock approach, the architecture chosen. The first project, KSOS, had a secure architecture but very poor performance. Although KSOS was a slow system, the architecture it is based on, TCB augmentation using trusted subjects, does not contain inherent performance problems.

The Sybase Secure SQL Server is an MLS DBMS that has been built based upon the addition of a trusted subject, viz. the DBMS server, to an existing TCB. The DBMS enforces its own access control policies for DBMS security objects, augmenting the security policy enforced by the operating system for its objects.

The Secure SQL Server enforces a security policy that is based upon the TCSEC endorsed Bell and LaPadula model. This policy contains both discretionary and mandatory components and applies to both user and system data. The discretionary access control policy applies to databases and tables. The owner of a database can grant and revoke permission to perform updates, selects, inserts, or deletes on that table to other users.

The basic unit of mandatory access control is the row. A user may select a row only if the user's login security level dominates, i.e. is greater than or equal to, the security level of the row. Rows are inserted at the user's login security level.

The database is stored in a collection of operating system managed files. These storage objects are protected by the operating system in several ways. Discretionary access controls are used to ensure that only the Secure SQL Server process has access to the operating system objects that contain the database. Mandatory access control is also used either by labeling the operating system objects at the highest level at which data might be stored within the database, or by allocating a special compartment for these objects and the Secure SQL Server process. Finally, the machine on which the Secure SQL Server runs can be configured as a database machine. This means that no users other than trusted system administrators are allowed access to this machine.

As was explained before for KSOS, Sybase implements an inherently secure architecture. Evaluation of the new composite TCB may be helped by a previous evaluation of the original TCB. Too much attention should not be focused on how much of the original TCB needs to be reevaluated. Evaluation is a lengthy but one time process. If a system is secure and easy to evaluate, but doesn't contain the other crucial user needs, performance, technology, portability and extensibility, the system will fail.

As was described for Seaview, modifying a pre-existing DBMS can be a significant challenge. The amount of effort required to modify a DBMS depends upon how closely the DBMS's internal structure matches the TCSEC requirements at the trust level being targeted. In addition, security features and assurances need to be added to the DBMS, such as MAC and auditing.

The trusted subjects approach allows the Secure SQL Server to extend the security policy of the base TCB. The DBMS security policy operates on a new set of subjects, objects and access rights. This is a powerful mechanism which allows the security policy of the overall TCB to be tailored to a specific application, in this case a trusted DBMS.

Performance of non-secure DBMSs was improved by having DBMSs provide their own resource management, instead of relying upon the resource management provided by an operating system. Similar gains can be achieved if the DBMS assumes responsibility for its own security policy enforcement, instead of relying on the underlying operating system. The trusted subject architecture gives the DBMS complete control over its security policy and resource mangement, therefore allowing for optimal performance.

Conclusion

When building trusted applications there are many factors which must be taken into consideration. The most crucial factor is the security of the system. Other important concerns include how easy the system is to build, evaluate, port and extend. The system must also contain state of the art technology and high performance.

For many years it was assumed that the best approach toward building trusted systems was to place the majority of the security features and assurances in a security kernel. KSOS, Trudata and Seaview took this approach, with varying degrees of success.

The trusted subject approach, adopted by Sybase, maximizes the security policy enforcement role of the DBMS. This approach allows the DBMS to manage its own resources, a necessary ingredient for high performance.

References

[1] "Department of Defense Trusted Computer System Evaluation Criteria," Dept. of Defense, National Computer Security Center, Dec. 1985.

[2] "Trusted Database Management System Interpretation," Dept. of Defense, National Computer Security Center, March 1990.

[3] Knode, Ronald B. and Roger Hunt, "Making Databases Secure with TRUDATA Technology," The Proceedings of the Fourth Aerospace Computer Security Applications Conference, IEEE, December 1988.

[4] Lunt, Teresa and R. Alan Whitehurst, "The Seaview Formal Top Level Specifications", SRI International, Menlo Park, Feb. 1989.

[5] McCauley, E.J. and P.L. Drongowski, "KSOS - The Design of a Secure Operating System," AFIPS Conference Proceedings, Vol. 48, 1979.

[6] Stonebraker, M., "Operating System Support for Database Management," Communications of the ACM, ACM, July 1981.

[7] Winkler-Parenty, Helena, "Can You Trust Your DBMS?," Database Programming and Design, July 1989.

AUTOMATED RISK EVALUATION SYSTEM (ARES)/
COMMUNICATIONS-COMPUTER SYSTEMS SECURITY MANAGEMENT SYSTEM (CMS)

Lt Glyn M Runnels
AFCSC/SRE
San Antonio, TX 78243-5000

INTRODUCTION

The backbone of the Air Force Communications-Computer Systems Security Program is the requirement to complete a formal risk analysis before a computer system is approved to operate. Air Force Regulation (AFR) 205-16, "Computer Security Policy", requires a risk analysis be performed in support of the accreditation of a computer system and the reaccreditation of that system every three years or upon major system change. In the past, though a considerable number of man-hours was expended in the risk analysis and accreditation of a system, little or nothing was done with the data collected beyond the end-user level. This wealth of information was either discarded or simply lay unused after the accreditation. A central collection capability was needed to make use of this data at the Major Command (MAJCOM) and even Air Force and Department of Defense (DOD) levels. Tracking even just a small fraction of the data collected would enable the Air Force to react more efficiently to incidents involving Air Force computer systems and to plan the development of the countermeasures to those incidents and their related vulnerabilities. With limited communications-computer security resources, this capability is needed now more than ever.

The Department of Defense established the Computer Security Technical Vulnerability Reporting Program (CSTVRP) to maintain a central repository of vulnerability information. The National Information Security Assessment Center (NISAC) is the DOD agency responsible for instituting the CSTVRP as directed by DOD Instruction 5215.2, "Computer Security Technical Vulnerability Reporting Program (CSTVRP)", 2 Sep 86. The instruction applies DOD-wide, including the Office of the Secretary of Defense, the Military Departments, the Joint Chiefs of Staff, all Commands, and all Defense Agencies.

To fulfill the Air Force requirement for the CSTVRP, the Air Force Cryptologic Support Center (AFCSC) is implementing the Communications-Computer Systems Security Vulnerability Reporting Program (CVRP). A major portion of the CVRP is the collection of accreditation data, the identification of vulnerabilities, and the notification of and development of countermeasures for those vulnerabilities. The CVRP Data Base, located at AFCSC, San Antonio, TX, will include an accreditation data base and a vulnerability data base. Data collected in the vulnerability data base will be forwarded to NISAC.

To fill these two data bases, AFCSC is developing two tools to collect and manipulate accreditation data from the end user to the Air Force level. The basic tool to be used by all computer system managers at all levels is the Automated Risk Evaluation System (ARES). At the MAJCOM level, a more restrictive software package will be used to control the data flow from the end users along the reporting chain to AFCSC. This software package is the Communications-Computer Systems Security Management System (CMS).

The basic package, ARES, is a risk management tool used to help maintain the computer system manager's environment and to assist in performing the duties necessary to accredit the computer system or systems. The software package is a menu-driven program consisting of several modules relating to the identification and use of the computer system. It produces reports on this identification and use, and also on areas which may require improvement and/or clarification along with letters relating to the accreditation of the system. It will provide a quick access source of data on the computer system manager's environment in a software format, enabling the storage of large amounts of information in a relatively small area. The primary users of ARES will be the Terminal Area Security Officers (TASOs) and the Computer System Security Officers (CSSOs), those at the management level closest to the end users.

The management tool, CMS, will collect data from the ARES users, collate it, analyze it, and present it to the CMS user in an easily understandable format. Currently, its primary purpose is the collection and collating of accreditation data and will be the primary source of the Air Force Accreditation Data Base. Besides data collection, its current functions include its own configuration management software, the ability to produce reports describing the security posture of a unit, a MAJCOM, or the Air Force, and vulnerability report management and notification. MAJCOM Computer System Security Managers (MCSSMs) and their counterparts in Separate Operating Agencies (SOAs) and Direct Reporting Units (DRUs) will be the users of CMS. Within each MAJCOM, SOA, and DRU, a central office is established to approve the accreditation of computer systems. Those offices generally delegate the approval process to other offices throughout the command; the management within those offices are the Designated Approving Authorities (DAAs). DAAs may use CMS if the amount of data being gathered requires CMS be delegated to that level. The relationships between ARES, CMS, and the levels of use will be explained later in this document.

Throughout this document, every use of MAJCOM will also include SOAs and DRUs. Similarly, MCSSM will also include the SOA and DRU counterparts, and CSSOs will also include TASOs. References to the DAA-level will mean the level of accreditation responsibility directly below the MAJCOM.

ARES

ARES input is interactive through many windows-like menus and requires only a keyboard; no other pointing device is needed. High level options include the type of report, editing and printing the system and safeguard modules, and printing reports. To obtain the widest user base, its hardware platform is a personal computer operating in a Disk Operating System (DOS) environment with 640KB of memory and a hard disk drive system.

After selecting the type of report to be generated, the system modules are selected. These modules describe the local environment and include IDENTIFICATION, SYSTEM USAGE, and NETWORKS information. IDENTIFICATION includes responsible officials, a mission statement, and descriptions of the system use, facts and assumptions, residual risks, and an overall conclusion. These sub-modules all use free form text input.

SYSTEM USAGE is an overall description of what the system is (type, environment, sensitivity/criticality), how it is used (clearance, classification, security mode, certification level), and to what degree it is used (information content, dates/times of operation, access). A description of the system in terms of the computer model, operating system(s), and security software is included, in addition to listing of other hardware, software and communications security equipment utilized. Safeguard modules are also selected in this module.

NETWORKS is a brief section describing connectivity with wide and local area networks and Host-To-Host and Host-To-Terminal connections.

Safeguard modules, selected from the SYSTEM USAGE section, are areas of potential risk. Some of the topics include audit trails, badges, housekeeping, and passwords. Based on the user's environment, the user selects options within each topic by toggling with the spacebar. These responses are processed by the program to produce a risk assessment for each safeguard module selected. For example, if housekeeping is selected in the SYSTEM USAGE module and the sub-module on dust control is not completed, the risk assessment portion of the final report will describe which areas of dust control must be corrected to comply with AFR 205-16.

ARES produces two forms of output. The first, the fact file, is the data file for the particular accreditation being performed. The data in the fact file is dependent upon the modules selected, as mentioned in the preceding paragraph. This data is stored in an ASCII file with the information from the modules stored in a specially delimited format. Basically, the format is

fact($*module_name*$,$*information*$**)**.

where *module_name* is the module selected by pressing the **<RETURN>** or **<ENTER>** key or the topic within that module, and *information* is either free form text keyed in from the IDENTIFICATION module or selections toggled using the space bar. Where several selections are possible, square brackets will delimit the data (*information1*, *information2*, etc.) from the *module_name*. This format is

fact($*module_name*$,[$*information1*$,$*information2*$,...]**)**.

Fact files are distinguished in the directory of the disk by the .FCT extension on the file name.

The second form of ARES output is the report. The standard ARES report is produced at the **COMMANDS** menu under **Print Report**. The data from all of the modules selected is collected from the .FCT file into this report and includes two cover letters, a cover sheet, the risk analysis report for the safeguard modules selected, and an operating guide based on those same modules. A report name is selected at the **Print Report** option and has a .RPT extension. The report is a pure ASCII file residing on the path defined within ARES, and it may be printed using the DOS PRINT command. In many cases, the report is imported to a text processor and edited to conform to the local and DAA requirements. This does not affect the .FCT file and the data gathered.

Individual module reports may also be generated while ARES is operating. Reports on the individual system and safeguard modules are selected from the **SELECT TOPIC** screen. For the system modules, IDENTIFICATION and SYSTEM USAGE are the only modules available for module reports; all safeguard modules may have reports created.

ARES POST-PROCESSOR (APP)

The APP is a small menu-driven program designed to extract data from ARES fact (.FCT) files. When run, the APP prompts the user to input a drive for the location of the fact files, and then provides a list of fact files found on that drive. After a choice of fact files is made, the APP proceeds to extract the required data from the file. After the initial pass through the fact file, the APP prompts the use for any needed data not found in the file. When completed, the output from the APP is an accreditation data file (ADF) with the same file name as the fact file but having a .ADF extension. This file is ASCII and comma/quote mark-delimited and is directly readable by CMS. Additionally, a receipt is printed to be returned to the ARES user showing the data extracted from the .FCT file. Also, where the .FCT file may be in excess of 12KB in size, the .ADF is generally less than 1KB.

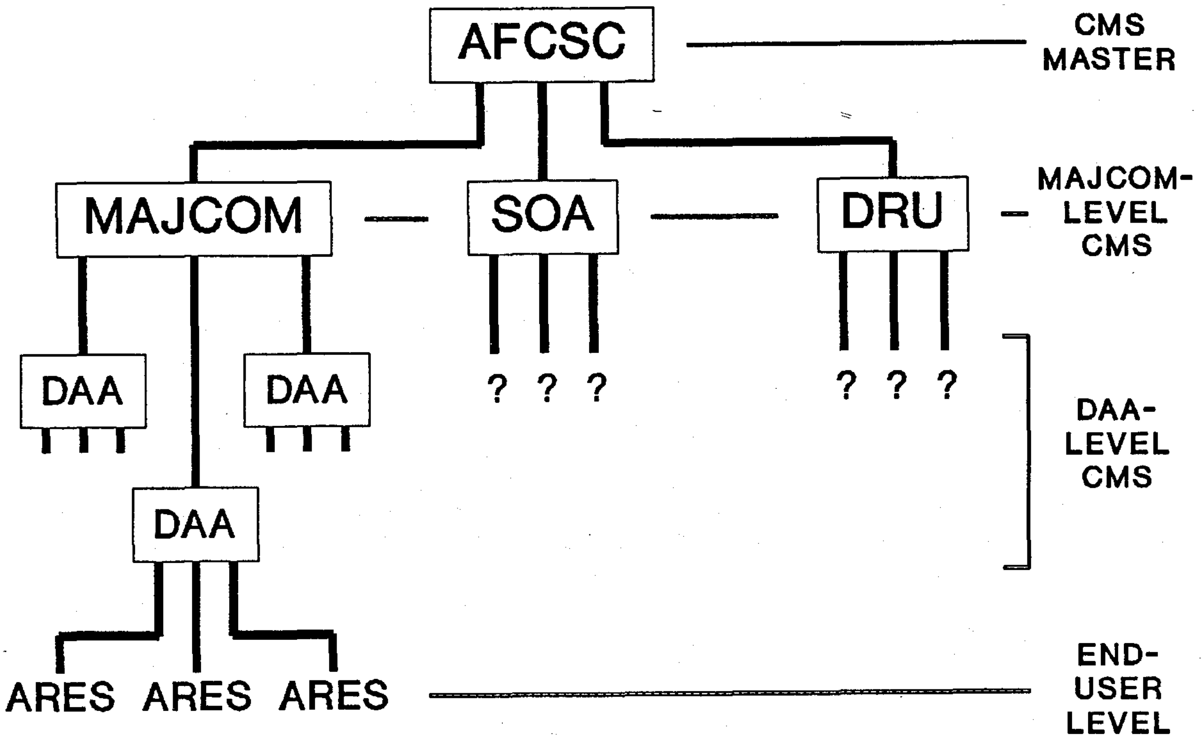
CMS

ARES data, in the form of .ADF files, is forwarded from the ARES users to the next level in the reporting chain. In some cases, the next level will be the MAJCOM; in most cases, the CSSOs will report their data to their DAAs due to the sheer amount of data handled. Smaller MAJCOMs and most SOAs and DRUs will probably be able to collect their data with the MAJCOM-level version of CMS; other MAJCOMs will collect their ARES data by having it passed through the DAA-level version of CMS. Figure 1 briefly illustrates this concept. Like ARES, CMS operates on a personal computer in a DOS environment with at least 640KB of memory and a hard disk drive system.

As shown in Figure 1, the master version of CMS will reside at AFCSC and will be the core of the Air Force Accreditation Data Base. The next level of CMS will originate at AFCSC through the use of the configuration management software. If it is deemed necessary, the MCSSM will use the same software module to create DAA-level versions of CMS.

The primary purpose of CMS is the collection and collating of accreditation data. One use for this data will be the reporting of vulnerability information. By far, the majority of the vulnerabilities discovered to date are resident in either the central processing unit (CPU), the operating system for the CPU, or the security software protecting the system. Vulnerabilities are usually discovered in one of three fashions:

1. Hacking incidents: someone breaking into a computer system through the use of another computer system
2. Malicious logic: viruses, worms, Trojan horses, or harm-causing logic which does not fall under the first three categories
3. Accidental but "friendly" discovery.



ARES/CMS Levels of Operation
Figure 1

The first two exploit vulnerabilities, or "holes", in the operating system, security software, or CPU, and can and have wreaked havok on computer systems, sometimes on a national scale. If a vulnerability has to exist, the third method is by far the preferred way of finding it; hopefully, this person or organization will report the vulnerability.

As a vulnerability is discovered, it becomes necessary to notify other users of similar hardware and/or software of the "hole" and to disseminate the required fix(es). These fixes can take the form of policy or direction, AFCSC-developed countermeasures, corrections to the hardware/software by the original contractor/developer, or any combination of these. Regardless of where the fix is developed, notification is necessary, as is the requirement to direct the implementation of those fixes first to those who have the greatest need.

The Accreditation Control Number (ACN) will provide a major link between AFCSC and end users for the notification and distribution of fixes. At this point, a discussion of the ACN and its assignment is necessary to understand the need to use the configuration management module.

ACN

The ACN is an eight-digit incrementing number assigned by the CMS software to each accreditation as it is loaded into the CMS data base closest to the CSSO. To avoid replication of ACNs, each CMS package will have a serial number imbedded in it by the next higher level of CMS, and this serial number will be part of the ACN. The assignment procedure follows.

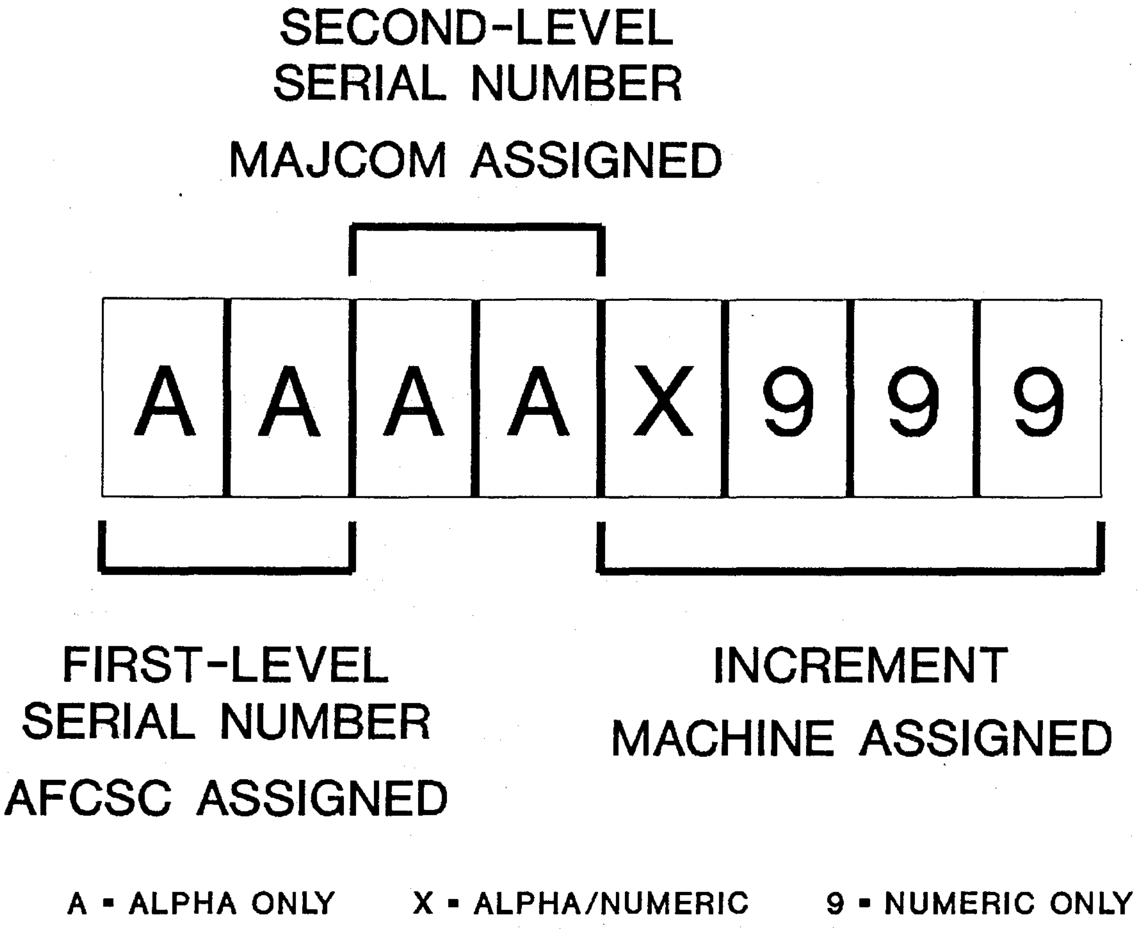
When the MAJCOM-level CMS software is created at AFCSC, a two-digit alpha code representing that MAJCOM is imbedded into the program by the configuration management software module. In addition, the MAJCOM-level package is configured to allow or prevent that level from making additional packages. For example, AFCSC makes a copy of CMS for MAJCOM X and the serial number "AA" is assigned to it. At the same time, by MAJCOM X's request, MAJCOM X's CMS software is allowed to make copies for levels in the reporting chain below MAJCOM X. MAJCOM Y, on the other hand, deems it unnecessary to produce lower-level copies of CMS, and its ability to produce those copies is inhibited. MAJCOM Y's serial number would be "AB". Any ACNs produced by X and Y would start with "AA" and "AB" respectively.

When the MCSSM is authorized and creates a DAA-level version of CMS, a four-digit alpha code representing that MAJCOM and the DAA-level of user is imbedded into the software, similar to the previously-cited example at the MAJCOM-level. The first two digits are those assigned to the MAJCOM by AFCSC; the second two are assigned by the MAJCOM to the DAA-level. For example, if MAJCOM X assigns serial number ZZ to an organization one level below it, the serial number imbedded into the DAA-level software will be "AAZZ". Similarly, all copies of CMS made by X will have "AA" and the two-digit alpha identifier for each copy imbedded in them. Due to the structure of the ACN, only one level of CMS below the MAJCOM may be created. This still gives the MAJCOM the ability to create over 650 copies of CMS for the DAA-level.

If the MAJCOM-level version of CMS produces an ACN, the ACN's first four digits will be its serial number followed by two zeroes. The ACN produced by the DAA-level of CMS will have as its first four numbers the serial number imbedded in it by its MAJCOM. The last four digits will be an incrementing number, starting with 0001 to a maximum of Z999. The highest incrementing digit is numeric from 0 to 9 and alphabetic from A to Z. Letters I ("eye") and O ("oh") are deleted to prevent mistaking them for 1 (one) and 0 (zero), respectively. This combination allows for a maximum of almost 34,000 accreditations per lowest level of CMS user. The ranges of ACNs for each level is as follows:

AFCSC (Master CMS):	00000001 - 0000Z999
MAJCOM-level:	@@000001 - @@00Z999
DAA-level:	@@&&0001 - @@&&Z999

@@ = Two-digit alpha serial number assigned by AFCSC to MAJCOM-level
&& = Two-digit alpha serial number assigned by MAJCOM to lower levels
Figure 2 shows the structural breakout of the ACN.



ACN Structure
Figure 2

ARES-CMS DATA FLOW

Accreditation data, in the form of ARES .FCT files, is collected by the CSSOs at the end-user level. When the data is forwarded to the next higher level, probably the DAA, it will be in one of two forms depending of the location of the APP software. If the DAA elects to issue the APP with the ARES software, the CSSO will send .ADFs to the CMS user. Otherwise, .FCT files will be sent forward where they will be converted to .ADFs before being entered into CMS. Currently, the files, .FCT and/or .ADF, will be sent up on floppy disks. In the future, other electronic means are anticipated, such as direct on-line upload and electronic mail.

At the lowest CMS level, the .ADFs are loaded using the CMS data collection software. As the .ADFs are accepted into the data base, the software prevents the loading of duplicate ACNs. If duplicates are received, the program prompts for instructions, such as replace, discard, etc. This process initiates the local accreditation data base.

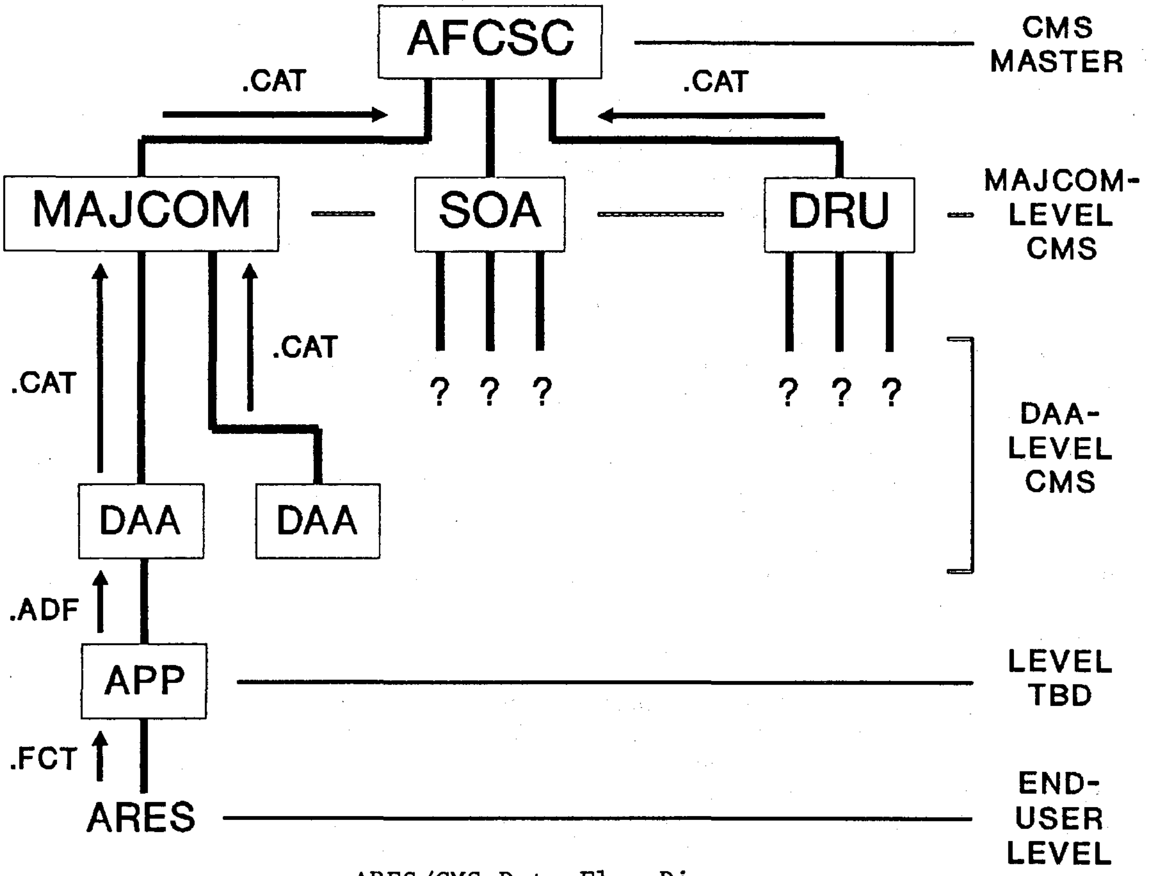
Periodically, the accreditation information will be forwarded up the reporting channels to AFCSC; currently, the intention is for a quarterly reporting period. The information forwarded will not be .ADFs but will be concatenated data files, or .CAT files. These .CAT files will be a product of the CMS software and will contain all the information from the .ADFs in the accreditation data base in a concatenated form. These .CAT files are directly readable by the next level of CMS.

If the lowest level of CMS is the DAA level, it will forward .CAT files to the level above it, the MAJCOM level. The MAJCOM-level version of CMS will collect the data from the many .CAT files from the lower level users and fill its accreditation data base. The information forwarded to AFCSC from the MAJCOMs will also be .CAT files with data from their accreditation data bases. Figure 3 shows the data flow from the ARES users to AFCSC.

TRACKING/NOTIFICATION

As a vulnerability is discovered, through one of the three means mentioned earlier, it must be tracked, notification of the vulnerability must be made as appropriate, a "fix" must be developed for it, and the "fix" must be distributed as needed. CMS, through data collected and distributed by the ARES users, will provide the mechanism for these actions.

There are several methods of notification available for reporting a vulnerability depending on the severity and urgency of the vulnerability. In



ARES/CMS Data Flow Diagram
Figure 3

most cases, regardless of the urgency, the MAJCOM will be part of the reporting chain. The Vulnerability Reporting module in the CMS software will provide the guidance necessary for reporting the vulnerability to AFCSC and, hence, to NISAC. The module will collect the necessary reporting information on the CPU in use, the operating system, the security software, the management personnel responsible for the system, etc., and forward this information to AFCSC. Due to classification issues, the actual description of the vulnerability will be sent separately.

Since all accreditations will be assigned an ACN, the vulnerability will be matched to its corresponding ACN and that record will be accessed in AFCSC's accreditation data base. The Vulnerability Reporting module at AFCSC will access all records in the data base with hardware/software configurations matching the reported configuration and provide a report to the CMS user. The report will be a listing of systems with similar configurations and will be sorted by criteria established by the CMS user. Primarily, the sorting will be from most sensitive and critical to least sensitive and critical; the CMS user will be able to further differentiate this list. Using this listing, AFCSC will be able to notify other users of similar configurations to the current or potential vulnerability. This reduces the necessity of notifying every user of every system concerning a problem which, in many cases, will not affect them.

Since it is very probable that several users may find the same vulnerability, CMS also accounts for this. As each vulnerability report (VR) is received, the module will assign a Vulnerability/Incident Control number (VIC). The VIC will be the Julian date, the MAJCOM serial number, and a sequence number. When VRs are received, the data is checked against existing VRs; if the information matches, an alphabetic suffix is appended to the existing VIC. This method allows tracking of VRs with less chance of duplication. In addition, as all fixes and/or countermeasures are applied and the VR is closed out, the VIC is closed also. The VIC and the related data are assigned to a history data base for future reference.

THE FUTURE OF ARES AND CMS

The functions and operations of ARES and CMS mentioned here are only an overview of their total capabilities. A major rewrite of ARES is in progress adding considerable functionality and converting to a new language. Some of these functions will be a local data base, graphics capabilities for modelling the user's environment, and enhanced TEMPEST and COMSEC options. Also, the functions of the APP will be absorbed into ARES, allowing ARES to directly produce both the .FCT files for local use and the .ADFs to send to the CMS users. ARES is not and will not be solely an accreditation package, but will continue to be a tool to help the CSSO manage the computer security environment more efficiently.

CMS will continue to be a modular software package. Modules being designed now include Security Resource Planning and a Security Forum. The Security Resource Planning module will provide the CMS user support in performing cost/benefit analyses to determine the resource impact of proposed countermeasures when a VR is determined to have command-wide impact. A general security information bulletin board is under consideration at AFCSC. The Security Forum module will provide dial-up communications capabilities to that and other approved boards and communications functions. Also, this module will provide the electronic mail and accreditation data upload capabilities mentioned earlier. When fully functional, this module will greatly reduce the time needed to update the accreditation information Air Force-wide.

ARES, CMS, and the program they both support, CVRP, are and will continue to be organic in nature. All three programs respond to the users' needs by evolving and adding functions and capabilities to support those users. They are here to make an already difficult job, accreditation, easier, and to benefit the Air Force and DOD by collecting, collating, and analyzing the information being gathered into a cohesive unit, and using this information to more efficiently utilize our increasingly limited security resources.

ACRONYM LIST

ACN	Accreditation Control Number
ADF	Accreditation Data File
AFCSC	Air Force Cryptologic Support Center
AFR	Air Force Regulation
APP	ARES Post Processor
ARES	Automated Risk Evaluation System
ASCII	American Standard Code for Information Interchange; a standard code adopted to facilitate the exchange of data between different data processing and data communications equipment
CMS	Communications-Computer Systems Security Management System
CPU	Central Processing Unit
CSSO	Computer System Security Officer
CSTVRP	Computer Security Technical Vulnerability Reporting Program
CVRP	Communications-Computer Systems Security Vulnerability Reporting Program
DAA	Designated Approving Authority
DOD	Department of Defense
DOS	Disk Operating System
DRU	Direct Reporting Unit
FCT	ARES FaCT file
KB	Kilobyte (1024 characters of storage)
MAJCOM	Major Command
MCSSM	MAJCOM Computer System Security Manager
NISAC	National Information Security Assessment Center
SOA	Separate Operating Agency
TASO	Terminal Area Security Officer
VIC	Vulnerability/Incident Control number
VR	Vulnerability Report

A TRUSTED SOFTWARE DEVELOPMENT METHODOLOGY

John Watson

GE Aerospace
Strategic Systems Department
1250 Academy Park Loop / Suite 209
Colorado Springs, Colorado 80910

Edward Amoroso

AT&T Bell Laboratories
Whippany Road
Whippany, New Jersey 07981

Abstract

This paper describes a Trusted Software Development Methodology that reduces the potential for subversion of Strategic Defense System (SDS) software during the system's design and development phase. The objective of the methodology is to identify measurable criteria and concrete procedures for determining classes of *trust* and guidelines for increasing the *trust* of new and existing SDS software. The methodology focuses on providing information to aid the SDS software manager, software developer, and software evaluator. This information provides guidance on how to integrate safeguards and countermeasures into the software design and development process in order to address specific threats and vulnerabilities.

INTRODUCTION

The "confidence value" of software is determined by the degree to which it possesses a desired combination of attributes that are critical to the success of the software's mission. Many factors determine the "confidence value" of software. Efforts in the field of Software Quality have resulted in guidelines and metrics for developing and measuring factors such as maintainability, reliability, performance, etc. [1]. In addition, efforts in the field of Computer Security resulted in mature guidelines and metrics for developing and measuring the strength of protection mechanisms that enforce confidentiality in an operational computer system. In the Strategic Defense System (SDS), the "confidence value" of software that performs mission-critical functions will be determined to a large extent by the *trust* of the software. Trust has attributes that are comparable to quality and security attributes. However, trust is focused on reducing the potential for software subversion throughout the system's lifecycle.

The degree to which software meets its requirements without containing exploitable flaws or malicious functionalities reflects the trust that may be placed in the software. A current means of imposing security and trust requirements and quantifying the resulting trust for SDS software is the "Trusted Computer System Evaluation Criteria" (the Orange Book) [2] and its associated guidelines and interpretation documents produced by the National Computer Security Center. However, the Orange Book primarily deals with disclosure-oriented threats during the operations phase of the system's lifecycle. SDS must also counter integrity and denial of service threats as well as disclosure threats. Moreover, these threats may be perpetrated during all SDS lifecycle phases. In order to counter these threats, a methodology is needed for integrating the following two processes into the software lifecycle:

- Impose requirements to ensure that the SDS is able to counter threats during its development and operation.
- Ensure that all software requirements affecting the security of the SDS, i.e., requirements which affect SDS's disclosure, integrity, and denial of service properties, are implemented without the introduction of exploitable flaws or malicious functionalities.

A Trusted Software Development Methodology has been created to address this need during the software design and development phase of the SDS lifecycle. The scope of this effort is defined by the collection of activities and products defined by the "Military Standard for Defense System Software Development" (DoD-STD-2167A) [3]. The components of the methodology are summarized below and described in the following paragraphs.

a. A precise definition of *software trust*.

b. An analysis of threats to software and vulnerabilities of software during design and development.

c. A flexible framework which allows for the uniform selection and organization of trust principles and methods of satisfying the principles.

d. A set of trust principles that specify requirements for safeguards and countermeasures during the design and development process.

e. Guidelines for modifying current development practices and environments so that they adhere to the trust principles.

f. Trust classes that define sets of trust principle requirements necessary to achieve an integrated approach to safeguarding against and countering threats that are targeted at exploiting software development vulnerabilities.

g. Trust measurement guidelines that are based on establishing compliance of trust methods with a set of trust principles that represent a target trust class.

h. Guidelines for maintaining and increasing the trust of existing software based on the capabilities and limitations of applying the trust principles defined by a given trust class or subclass.

DEFINITION OF SOFTWARE TRUST

The foundation of the Trusted Software Development Methodology is a baseline definition of software trust and the specification of primary trust properties. The definition shown below is intended to capture the major properties of software trust and provide a framework for trust measurement, development, and retrofit guidelines.

> *Software trust* is the quantification of evidence that the lifecycle process used to create and transform software: 1) yields a product that exactly satisfies specified requirements, and 2) counters the threats targeted at the software.

The primary properties of software trust are summarized below:

a. The trust quantification method is based on objective facts and it always produces the same value for a given input.

b. Both process-oriented evidence and product-oriented evidence are used to establish trust.

c. The interpretation of evidence within the trust quantification method is performed with respect to a statement of vulnerability induced threats that directly affect the operational capabilities of the software.

d. A product that *exactly* satisfies specified requirements has *all and only* those elements necessary for conformance to requirements.

e. In application, software trust is a measure of how well the product meets the specified requirements, adjusted appropriately by the measure of requirements accuracy and completeness.

THREATS TO SOFTWARE DURING DESIGN AND DEVELOPMENT

The threat to SDS software is summarized in the following statement:

> **The general threat to SDS software is the undetected insertion of flawed or malicious software that has the potential to adversely affect the required functionality and/or information resources of the system in the form of destruction, disclosure, modification, and/or denial of service of the required functionality and/or information.**

The threat described above is typically referred to as *software subversion*. Subversion can be accomplished at any time during the system's lifecycle from the earliest stages of system specification to obsolescence and removal. Software can be subverted either to permit later penetration or as an act of sabotage -- to nullify or to degrade the system's capability. Forms of software subversion include Trojan horses and viruses. Requirements must be implemented in the SDS through software and security engineering in order to counter these threats of subversion.

The Trusted Software Development Methodology is designed to specify requirements and methods for safeguarding against and countering these threats as they relate to exploiting vulnerabilities of the software design and development process. As shown in Figure 1, the primary emphasis of the methodology is on preventing the introduction of malicious software that has the potential to cause loss of integrity or denial of service. The secondary emphasis of the methodology is on preventing inadvertent human errors that can result in loss of integrity or denial of service. Thus, the Trusted Software Development Methodology is an adjunct to many well established approaches that are intended to identify requirements and methods for countering threats to confidentiality.

THREAT OBJECTIVES		THREAT MEANS
Loss of Integrity	Highest Priority	introduction of malicious software
Denial of Service	|	inadvertent human errors
Compromise	Lowest Priority	penetration of data or communication processors and networks for the purposes of acquiring critical data

Figure 1. Threat Priorities for the Trusted Software Development Methodology

The characteristics listed below summarize the primary threat that is addressed by the Trusted Software Development Methodology. Secondary characteristics are identified in parentheses.

Target: All software products created and modified during the design and development process.

Perpetrator: Intelligent insider with malicious intent.

Objective: Loss of integrity (and denial of service).

Means: Subvert/modify software (and inadvertent human error).

Threat Class: Vulnerability-induced threats (and human error).

Period of Attack: The software design and development phase defined by DoD-STD-2167A.

VULNERABILITIES OF SOFTWARE DESIGN AND DEVELOPMENT

While threats may be identified and enumerated, the identification and characterization of system penetration vulnerabilities that support successful attacks is a far more complex and difficult problem. Vulnerabilities offer the opportunity for threats to be executed. Threats can only be carried out because hardware, software, or procedural deficiencies (i.e., vulnerabilities), make unauthorized accesses to system components possible. These vulnerabilities represent an inability of the software development system, execution environment, or maintenance system to restrict developer, operator, and maintainer accesses to only those components necessary to satisfy their needs and responsibilities. There are two basic categories of vulnerabilities:

- Vulnerabilities in the hardware and software components that permit an attacker to carry out a threat successfully.
- Unpredictable failures (including protection means) and human errors that permit attacks to be conducted.

In the first category, vulnerabilities in the software are fundamentally the access paths to system components that facilitate their misuse. They stem from design and implementation flaws introduced during system development and make the software susceptible to attack. Design and implementation flaws include both missing and inadequate protection mechanisms for preventing unauthorized access of system components. For example, a design flaw would be the failure to include a protection mechanism in a system for preventing unauthorized access. An implementation flaw would be the "misinterpretation" of a design specification that led to an operational malfunction and failure defect, eventually resulting in unauthorized access and subsequent misuse of system resources.

In the second category, failures in protection means and related software elements are probably the most common. Access control means in the software can fail and support either accidental or deliberate attacks. For example, errors in communications software may cause the misrouting of information and thereby support attacks. Thus, it is desirable to have fault detection, isolation, and automatic error recovery capabilities for software posing such vulnerabilities.

Based on the above characterization, the vulnerabilities of SDS software can be summarized in the following statement:

> **A vulnerability of SDS software is a weakness in design, implementation, testing, or development controls that could be exploited to insert undetected flawed or malicious software.**

SAFEGUARDS AND COUNTERMEASURES

By applying the appropriate specification, design, coding, testing, and integration techniques, the software development process and resulting products can have inherent protective attributes or built-in safeguards that make it hardened against threats. This is the premise of the Trusted Software Development Methodology.

An analysis of threats to software and vulnerabilities of software is required in order to identify specific vulnerabilities that could be exploited by the threat to software during design and development. The vulnerabilities can then be used to derive requirements for safeguards and countermeasures that can be implemented to reduce the threat–to or vulnerability–of software. Each requirement is represented as a *trust principle*. For each principle, there are alternate means of satisfying the requirement stated by the principle and, therefore, countering the threat. Each means is represented as a *trust method*. Hence, three basic conclusions result from an analysis of threats and vulnerabilities:

a. Software is a valued SDS asset with specific vulnerabilities that can be exploited by specific threats.

b. Trust Principles are requirements for protecting the asset.

c. Trust Methods are the means for satisfying the trust principle requirements.

Based on the threat and vulnerability characterization presented above, the following set of criteria was developed to validate the selection of trust principles and methods that represent strong safeguards and countermeasures for software trust:

Perpetrator: Reduce the potential for subversion/modification of software by an intelligent insider (developer or development support personnel) with malicious intent.

Target: Protect the integrity of tangible products resulting from the software design and development process (i.e. code, documentation, algorithms, designs, databases).

Period of Attack: Control, eliminate, or minimize software vulnerabilities during the system lifecycle's design and development phase.

Availability of the Target: Reduce the relative accessibility of software vulnerabilities.

Availability of the Means: Reduce the relative availability of methods that can be used to subvert/modify software.

Gain to Perpetrator: Reduce the relative gain of methods that can be used to subvert/modify software.

Detection: Increase the potential for detecting a subversion/modification of software.

Containment: Increase the potential for containing, preventing, or controlling a subversion/modification of software.

TRUST PRINCIPLES

Trust principles describe software development process characteristics that increase the trust in software. These characteristics define safeguards and countermeasures that decrease the potential for subversion during software development. Thus, trust principles are designed to specify requirements for an integrated, comprehensive protection approach for software during the software development lifecycle.

Trust principles can be applied in the software development lifecycle in two different ways. If a trust principle describes characteristics of a process that is already addressed in DoD-STD-2167A, then that trust principle defines a modification that must be made to the DoD-STD-2167A process. On the other hand, if a trust principle describes characteristics of a process that is not included in the DoD-STD-2167A lifecycle, then that trust principle defines an addition to the DoD-STD-2167A requirements. An example of such an addition is a collection of trust principles that refer to "verification." Since the only assurance activity that is mentioned in DoD-STD-2167A is testing, compliance with the DoD-STD-2167A requirements as well as the verification trust principles will require that the lifecycle be augmented with verification activities.

Trust principles are documented using a format that includes the following:

Name: A descriptive name for the principle.

Principle: A concise requirement statement of what needs to be done in order to control, eliminate, or minimize a specified vulnerability of software.

Vulnerabilities Addressed: Reference to the vulnerabilities addressed by the principle.

Rationale: A description of how the principle controls, eliminates, or minimizes the vulnerability.

Compliance: A description of the attributes that are required in order for a trust method to sufficiently satisfy the principle.

References: Publications that support the stated principle.

Trust Principle Selection

To create an appropriate set of trust principles, a set of principle selection requirements is needed. These requirements determine the validity of a trust principle. The attributes necessary for a trust principle are summarized below:

Requirement: The principle must be a necessary requirement for trust. There must be one or more clearly identifiable software vulnerabilities that are likely to exist if the principle is not satisfied.

No Alternatives: There must be no alternative principle that, individually, would sufficiently address the associated software vulnerabilities.

Consistent: The principle must be consistent with all other principles. A principle must not prevent application or reduce the effectiveness of another principle.

Practicable: There must be at least one clearly identifiable method for satisfying the principle.

Measurable: Adherence to the principle must be effectively determinable by the compliance criteria.

Technical Foundation: The principle must be based on a sound technical foundation and should be generally accepted in the software and security engineering communities.

Trust Principle Organization

When examining the processes and products of software development, it is apparent that trust principles can have various ranges of influence. These ranges of influence are shown in Figure 2. Some principles are enforced during all processes of the software development lifecycle, and hence, affect the trust of all software products. These principles characterize the software development environment, tools, and procedures that must be in place throughout the lifecycle. Application of these principles provides a uniform approach to safeguarding and countering vulnerabilities of software throughout the lifecycle, both during execution of software activities and during the transition from one activity to the next. An example of this type of principle is shown below.

Least Privilege Principle: The software development environment and tools shall provide a means for enforcing least privilege for each software developer.

Other principles are specific to a given process, and their influence is limited to the products resulting from that process. Application of these principles provides safeguards and countermeasures that are applied to protect the products resulting from the given process. An example of a principle that is specific to the coding process is shown below.

Code Analysis Tools Principle: All software shall be provided as input to static analysis tools in order to identify coding techniques that are overly complex or that contradict standard quality enhancing approaches.

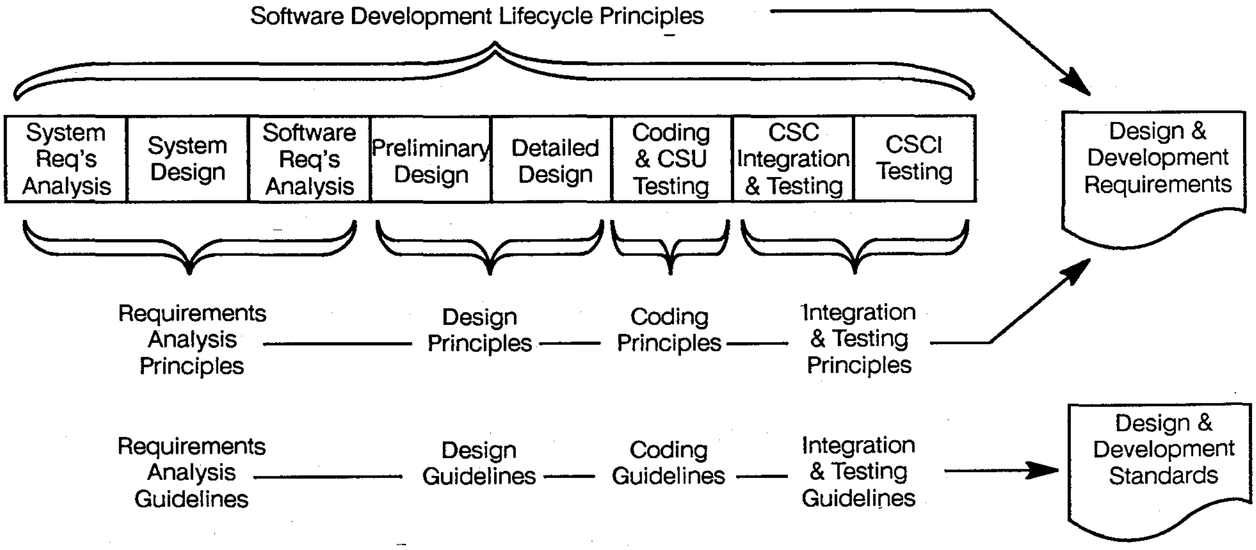

Software Development Lifecycle Principles
System Req's Analysis
System Design
Software Req's Analysis
Preliminary Design
Detailed Design
Coding & CSU Testing
CSC Integration & Testing
CSCI Testing
Design & Development Requirements
Requirements Analysis Principles
Design Principles
Coding Principles
Integration & Testing Principles
Requirements Analysis Guidelines
Design Guidelines
Coding Guidelines
Integration & Testing Guidelines
Design & Development Standards

Figure 2. The Ranges of Influence For Trust Principles

In all processes of the software development lifecycle, there are varying levels of requirements. There are high-level requirements such as the trust principles shown above, and there are low-level requirements that are application- and environment-specific. The high level requirements typically allow for some flexibility in their implementation. A developer examines the collection of high level requirements and proposes methods of meeting the requirements in a manner that is consistent with a variety of development parameters.

In order to ensure that requirements will be satisfied in a consistent and uniform manner, the developer generates standards for various activities. These standards are compiled from a combination of existing and new guidelines that have been selected specifically for their applicability in enforcing a consistent and uniform implementation of the high level requirements. The Trusted Software Development Methodology includes a Standards Principle requiring that *no trust-critical operation shall be committed unless it conforms to an explicitly-stated project-defined design and development standard.*

Standards are typically created from a combination of guidelines for achieving required software attributes (i.e. reliability, maintainability, trust, etc.). For each major activity of the software development lifecycle there is a trust principle requiring that the activity be performed in accordance with a set of detailed trust guidelines that can be used to augment existing guidelines that support quality and security requirements. An example of a trust guideline for the Coding Activity is shown below.

> Avoid Pre-defined Names: When declaring programmer-declared items, avoid creating identifier names that are the same as those in the pre-defined language environment. Do not create identifiers that are the same as the names of 1) attributes or pragmas, or 2) items contained in package specifications for STANDARD, SYSTEM, UNCHECKED_CONVERSION,TEXT_IO, etc.

A detailed description of all trust principles and guidelines can be found in the "Trusted Software Report" [4].

TRUST CLASSES

Trust principles are grouped to provide integrated approaches to safeguarding and countering software vulnerabilities. These groupings are defined as trust classes. Adherence of a particular software development approach to a trust class is determined by establishing compliance of that approach with each of the trust principles in that class. Four classes of software development approaches are defined in relation to the groups of principles shown in Figure 2:

- *Comprehensive Class* represents a development approach that satisfies the total set of trust principles. This includes all Software Development Lifecycle Principles and the entire collection of principles that are specific to each software development process (i.e. Requirements Analysis Principles, Design Principle, etc.).

- *Uniform Class* represents a development approach that satisfies all and only the Software Development Lifecycle Principles.

- *Lifecycle Specific Class* represents a development approach that satisfies all and only the collection of principles that are specific to each software development process (i.e. Requirements Analysis Principles, Design Principle, etc.).

- *Basic Class* represents a development approach that may satisfy some principles in one or more of the other classes, but does not satisfy all principles in any one of the other classes.

These classes are arranged in a hierarchy according to their respective levels of trust. The hierarchy is based on the degree to which effective safeguards and countermeasures are specified by the trust principles of the particular class. Since it is more desirable to provide safeguards and countermeasures that span the entire lifecycle, including all lifecycle activity boundaries, rather than providing safeguards and countermeasures that are bounded by individual lifecycle activities, the following ordering is established on the set of trust classes:

Basic < Lifecycle Specific < Uniform < Comprehensive

The trust classes define a basic framework that can be tailored based on the requirement to address additional vulnerabilities that are of specific concern for a given environment or application. For example, it may be determined that the products resulting from the requirements analysis process need high levels of trust, since all software products in subsequent activities are derived from this process. The need to emphasize this process can be met by requiring a development approach that satisfies the Uniform Class and also satisfies the subset of Requirements Analysis Principles in the Lifecycle Specific Class. This approach is more trusted than just the Uniform approach. In this manner, intermediate subclasses can be defined between each of the four classes.

INCORPORATING TRUST IN THE SOFTWARE DEVELOPMENT PROCESS

The following paragraphs summarize the activities necessary to incorporate trust in the software development process.

Specify Trust Requirements: First, an appropriate set of software development trust requirements must be identified. These requirements are represented as a *required trust class*. The process of specifying a required trust class is based on predetermined guidelines for determining the minimum trust class required for a given software process or product. The required trust class is determined by examining the threat statement, vulnerabilities, mission-criticality, and security-critical requirements for a given software development process or product. This trust class can be augmented with additional trust principles that are not included in the target class, but that are of particular use in countering environment-specific software development vulnerabilities.

For each trust class, there is a corresponding set of trust principles and associated compliance criteria. Thus, the required trust class determines 1) the *required trust principles* that must be applied during software development in order to achieve the required trust class, and 2) the *compliance criteria* that must be applied during evaluation of proposed trust methods in order to assess the level of adherence to the required trust principles.

Identify Trust-Critical Software Operations: Many trust principles that span the entire software development lifecycle are based on monitoring and controlling trust-critical software operations. In order to satisfy the required trust principles, the developer must first determine which software operations are trust-critical.

For the majority of trust principles, determining the associated software development activity is straightforward. For example, a trust principle that stipulates the use of a validated Ada compiler is obviously associated with the activity in which a compiler is selected and used. Similarly, trust principles describing security testing, requirements analysis, prototyping, code verification, and so on, are easily associated with their corresponding software development activity. However, certain trust principles must be applied to a set of software development activities that are less obvious. For example, a trust principle that stipulates on-line auditing in the software development environment, requires that a set of auditable activities be identified. If this set includes "all" software development activity, then one could interpret this to include minor activities such as individual keystrokes during a working session. If, on the other hand, this set includes only a subset of software development activity, then the potential is introduced that important activities might not be audited. Thus, an appropriate set of software development activities must be identified for this type of trust principle. In order to perform this determination, the concepts of software operations and of trust-critical software operations are introduced below.

A *software operation* is any software development activity that produces a state change in a version of the software or its associated documentation that is under configuration management. Note that software operations are associated with "change." Thus, operations such as examining code or reading a document would not be considered software operations. Instead, the activities corresponding to insertion, deletion, or modification to a software version, are considered software operations.

Software operations cannot be considered separately whenever they consist of several different activities that could be interrupted, thus leaving the software in a state in which an operation is being performed, but has yet to complete. For example, software modification could require invocation of several software development functions in a sequence. If this sequence is initiated, but has yet to complete, then the modification operation is not complete. Software operations are considered complete when the operation is irreversible within the constraints of the procedures, protocols, and controls that exist in the software development environment.

A *trust-critical software operation* is any software operation that can be used to introduce or exploit a software vulnerability. All software operations are not considered trust-critical software operations simply because they are associated with "change." The relationship between software operations that are trust critical and software operations that are not trust-critical is analogous to the relationship between events that are auditable as defined in the Orange Book, and events that are not auditable.

Develop Trust Methods That Meet The Compliance Criteria For The Required Trust Principles: Trust principles describe *what* is required and trust methods describe *how* the requirements will be met. For each required trust principle, a corresponding trust method shall be proposed, evaluated for compliance with the trust principle, and approved prior to software development. Trust methods should be included as part of the developer's proposal or, as a minimum, included in the Software Development Plan. The template shown below describes the information necessary to appropriately describe a trust method.

Method: A concise statement of a recommended approach for satisfying the referenced principle and enforcing the method.

Principles Supported: A reference to the name of the principle supported.

Rationale: A description of how the method satisfies the principle (benefit of implementing the method). In other words, how does the method control, eliminate, or minimize the vulnerability addressed by the principle?

Evidence: Directly measurable characteristics of the software development process and/or product that represent objective, demonstrable evidence that the method is sufficiently implemented.

Impact: A description of various factors that affect the "cost" (effort, dollars, time, new tools, training, maintenance, etc.) of implementing the method.

For each application of the Trusted Software Development Methodology, the developer must determine a set of methods that both satisfy the required trust principles and are consistent with the parameters of the development environment and application. The optimum selection of trust principles is based on many factors that will vary from one developer to another. Existing tools, procedures, experience, development platforms, and quality and security requirements may all have an affect on choosing the optimal collection of trust methods.

When identifying alternative methods and their associated evidence, the burden of proof that a method satisfies a principle is on the developer, not the evaluator of trust. In order to assist in this determination, a set of requirements for method acceptance have been defined. Trust methods shall be evaluated against the following selection requirements.

Compliant: The method must directly correspond to the referenced principles. It must meet the compliance criteria that are required in order to satisfy the associated principles. It must sufficiently control, eliminate, or minimize the vulnerabilities addressed by the associated principles.

High Benefit: There must be a high benefit relative to the associated impact of implementing the method.

Consistent: The method must be consistent with all other trust methods of a proposed software development approach.

Practicable: There must be a practical means of implementing the method with currently available technology.

Measurable: There must be directly measurable characteristics of the software development process and/or product that represent objective, demonstrable evidence that the method is sufficiently implemented.

Apply An Integrated Set of Trust Methods: An integrated set of software development methods collectively defines the software development methodology. For SDS, this methodology is compliant with the requirements in DoD-STD-2167A. Trust requirements may necessitate modification of existing aspects of the proposed software development approach. Therefore, some development activities will have to be either added or modified.

The manner in which proposed software development methods comply with the requirements for both the trust principles and DoD-STD-2167A is straightforward in most cases because trust principles make the DoD-STD-2167A requirements more explicit. For example, the DoD-STD-2167A requirements for security (paragraph 4.1.5 in DoD-STD-2167A) are complemented by a trust principle that requires that the software development approach comply with an explicitly-defined security policy. The requirements for compliance with the trust principle define security policy characteristics that must be met during the software development lifecycle. Compliance with these requirements does not require an extensive change in the existing approach.

In some cases, the trust principles do require additions to the DoD-STD-2167A lifecycle. For instance, the software engineering environment detailed requirement (paragraph 4.2.2 in DoD-STD-2167A) is augmented by trust principles that require on-line auditing, access mediation, trusted path, and so on.

Measure Compliance With Trust Requirements: Compliance with trust requirements is determined by both the required trust principles and the implemented trust methods. The evaluator of software trust needs to know whether a given trust method sufficiently supports a trust principle and whether there is sufficient evidence that the method was properly implemented.

Each trust principle includes a description of the attributes that are required in order for a trust method to satisfy the principle. This is documented in the compliance field of each trust principle description. Thus, each trust method is evaluated by assessing the degree to which the method meets the compliance criteria associated with the principle.

Demonstrable evidence of adherence to trust principles is determined by trust methods that support the principle. During the software development lifecycle various evidences of software trust will be gathered and evaluated based on the predetermined evidences that were proposed and approved at the beginning of the development lifecycle. Evidence of acceptable trust method application may include key pieces of information that exist in reviews and audits of both products and processes of the development lifecycle.

TRUST RETROFIT

Trust retrofit is defined as the task of maintaining or increasing the trust of existing software. This definition presumes that a trust measurement has been performed to determine the level of trust to be associated with the existing software. In other words, the software development approach used to develop the existing software is examined with respect to all trust principles, and if all principles for one of the defined trust classes are met, then the software is consistent with that trust class.

For existing software, trust cannot be increased to a higher class or subclass by adherence to additional Uniform trust principles, because Uniform trust principles require adherence during the entire development lifecycle. Since, for existing software, the development lifecycle is either ongoing on complete, Uniform Class principles cannot be met. However, trust can be increased to a higher class or subclass in existing software by adherence to Lifecycle Specific trust principles. Adherence to Lifecycle Specific trust principles requires adherence only during a particular lifecycle activity. Thus, by repeating that activity, and all subsequent activities, trust can be increased.

The impact associated with such retrofit will vary. In general, impact increases as the lifecycle activity chosen for retrofit is earlier in the development process.

Since only Lifecycle Specific trust principles can be used for retrofit, certain bounds are placed on the retrofitting of existing software toward greater trust. These bounds are summarized as follows:

- Basic Software can be retrofit to Lifecycle Specific and all intermediate subclasses.
- Lifecycle Specific Software cannot be retrofit to greater trust.
- Uniform Software can be retrofit to Comprehensive and all intermediate subclasses.

Although the Uniform Class trust principles cannot be used to increase trust in existing software, they can be applied to maintain existing levels of trust. For example, in a software development effort that has already been initiated, if the Uniform trust principles have not been adhered to from the outset, then they cannot be adhered to for the entire lifecycle. However, initiating methods that support these Uniform Class trust principles will ensure that the present level of trust is not degraded further as the lifecycle progresses.

SUMMARY

This paper described a Trusted Software Development Methodology that is designed to allow prediction, measurement, feedback, and control of vulnerabilities in the SDS software development processes and products. The methodology provides a means of identifying the principles and criteria of software trust so that trust can be 1) included as an integral part of the development of a software product (trust development), 2) measured in an existing software product or process (trust measurement), and 3) retrofitted in existing software products (trust retrofit). The trust principles and criteria are initially applied in the prediction and measurement of trust. Trust measurement examines the presence, absence, or degree of identifiable software characteristics that are evidence of trust, and it yields a rating that indicates the relative level of trust achieved.

ACKNOWLEDGMENTS

This paper summarizes the first year's results of the INFOSEC Software and System Analysis Program contracted by the Strategic Defense Initiative Organization. A detailed description of the methodology is documented in the SDI "Trusted Software Report" dated June 29, 1990. The authors acknowledge the combined efforts of all team members from General Electric, AT&T, and Dynamics Research Corporation in contributing to the source material and reviewing this paper.

REFERENCES

[1] Thomas P. Bowen, Gary B. Wigle, and Jay T. Tsai, *Specification of Software Quality Attributes*, RADC-TR-85-37, February 1985.

[2] Department of Defense, *Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

[3] Department of Defense, *Military Standard Defense System Software Development*, DoD-STD-2167A, February 29, 1988.

[4] GE Aerospace, et al, "Trusted Software Report," SEIC CDRL Item A075, 29 June 1990.

A CATEGORIZATION OF PROCESSOR PROTECTION MECHANISMS

Eugene D. Myers
Office of Research and Development
National Computer Security Center

Abstract

Processor mechanisms useful for a trusted computer base (TCB) can be categorized by the degree of control placed upon a subject's access to objects and other domains. The mechanisms include those that enforce least privilege and the assists provided to an operating system. Operating system assists include primitive operating system functions such as process dispatching and synchronization. The protection mechanisms are categorized in terms of granularity, which is the degree to which the processor confines the actions of a subject. Granularity ranges from coarse to fine, where minimal confinement is coarse granularity and the maximum confinement is fine granularity. Confinement of a subject's actions has two aspects. The first aspect is the limitation of an active subject's access to other domains. The second aspect is the limitation of a subject's access to its own domain or address space, which is reflected in how distinguishable individual objects are to the subject. Operating system assists and protection mechanisms become increasingly related at finer granularity. This relationship is because at fine granularity the most privileged domain is not visible to the other domains and the operating system assists are located within the machine boundary.

1. INTRODUCTION

One issue in the design of a computer system is the protection of the data within that system. Data protection is more than access control. It also provides a measure of safety in preventing the corruption of data. There are many types of data within a computer system and each type is protected by a different access monitor. Two examples of data types are files and primary memory locations. The emphasis of this paper is on the part of the system that controls access to primary memory. The memory access checker and related features are called the processor protection mechanisms.

The processor protection mechanisms form the foundation for the rest of the protection features of the system. The protection mechanisms provide the primitive functions that are used to support the security policy of the system. The operating system uses these mechanisms to keep programs separated and to protect itself. This paper describes the theoretical basis for the mechanisms, the degree of control that can be attained, and finally the relation of current machine's mechanisms to the categories.

2. BACKGROUND

Processor protection mechanisms have a theoretical basis in computer security. Protection mechanisms constrain the actions of a program, and limit the program's privileges to the least necessary for the program to do its task. The system grants privileges on the basis of the security policy.

2.1 Domains, Objects, and Subjects

The smallest element protected in a computer system is called an object. Objects are entities that can be observed, manipulated, or activated by an executing program. In this paper, the objects of interest are those protected by the hardware and the ones that are accessed as if they are in primary memory. A domain is a set of objects related by an access function to another object, called a subject. An example of a domain is a process address space, which contains the data and executable code areas. Other objects that can be within a domain are the special control registers for the processor and devices, and the privileged instruction set. A domain is also an object, and the ability to transfer control to another domain is a privilege. A process is a logical processor which includes the processor registers, instruction counter, and process stack pointer. There can be more than one active process within a domain. Multiple processes within a domain, called lightweight processes or threads, are supported in contemporary operating systems[13]. A subject is defined as an ordered pair (process, domain)[7]. In other words, a subject is an object, referenced by the logical processor, whose contents are used by the processor as instructions, and whose domain is a collection of objects to which the subject has access rights. Finally, a subject's access rights to each of its objects can be different.

2.2 Access Matrix

The Access Matrix defines an abstract model of protection within a computer system[7]. Every time a subject attempts to access an object, the access matrix is consulted to determine if the access is permitted. The policy manager is invoked when a subject requests initial access to an object. The policy manager determines the subject's privileges (or access rights) based on the security policy, and places the privileges into the access matrix. The procedure of determining and placing a subject's privileges to an object is called access policy mediation.

An access matrix can be envisioned as a matrix with the subjects along the side and the objects along the top as shown in Figure 1. Each location in the matrix, denoted by (S_i,O_j), holds the privileges subject S_i has to object O_j. The types of access rights are system dependent. Some examples of access rights are read (r), write (w), append (a), and execute (e). Each location (S_i,O_j) is set according to the access policy when the subject activates or opens an object. When a subject attempts to access an object, the access matrix is checked to determine if the requested access is allowed. If the access request is denied, the system takes action to handle the offending subject. The run time checking of the access matrix is known as policy enforcement.

	O_1	O_2	. . . . .	O_n
S_1	r	w		
S_2	w			e
.				
.				
.				
S_M		r/w		e

Figure 1

2.3 Reference Monitor

A reference monitor is conceptually a single agent that enforces the authorized access relationships between the subjects and objects of a system[1] . It is a gatekeeper in the path between the subjects and objects. A reference monitor must meet three design requirements: 1) it must be tamperproof, 2) always invoked, and 3) always operated according to the security policy. If any of these requirements are not met, there will always exist the possibility of unauthorized access.

The realization of a reference monitor in a computer system distributes the policy enforcement rather than having a single enforcement entity. Graham and Denning[4] [4] depicted a model of distributed enforcement when they extended the access matrix model. The hardware and software protection mechanisms were generalized into what they called "monitors." The monitors represented the system intervention done at execution time when a subject accesses an object. Each monitor controls access to an object type by interpreting the access matrix. Examples of monitors are: a file manager that polices access to a file and a memory monitor, or memory management unit (MMU), that checks every access to primary memory. This paper concentrates on the monitor that checks memory access.[1] The use of monitors and an access matrix to enforce privileges implies a separation between policy and mechanism. The access matrix is itself an object where all monitors have read access and only the access matrix monitor has write access. The access matrix monitor acts as the policy mediator when updating the access matrix. Separating policy and mechanism improves trust, allows for the implementation of different policies,

[1] In some computer systems, the enforcement of file access and memory access is done by the same mechanism. For example, MULTICS treats secondary memory as an extension of primary memory. The memory monitor table is initialized when the file (or object) is opened and all checks are made in the same manner as accesses to primary memory. File I/O is handled by the paging mechanism which does not have any knowledge of the privileges associated with the object. Enforcement is done by the memory management unit. In a system that has traditional file I/O, such as UNIX, access rights must be enforced by the file system on every I/O request. For example, the file manager must disallow any writes to a file in which the subject has only read permission.

and improves efficiency. The policy element is centralized and the enforcement mechanism is distributed to the monitors of each object type. This separation provides a single consistent policy for the system, and the type monitors only have to be shown to correctly enforce the object privileges. Combining policy and mechanism makes it difficult for alternative policies to be implemented, or for the current policy to be updated. Efficiency is also a problem in combined policy and mechanism systems. If the policy has extensive rules, the computation of the access rights upon every access degrades the system performance. To support the separation of policy and mechanism, the machine must provide the mechanisms to support a broad range of security and integrity policies.

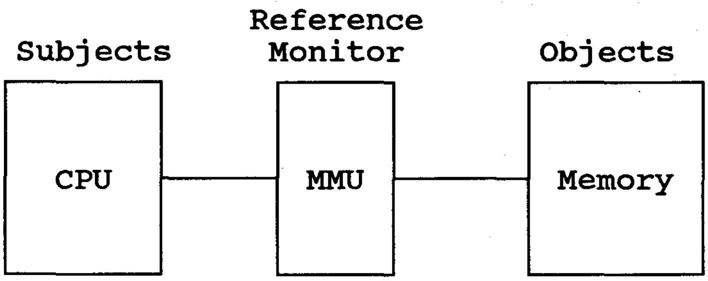


Figure 2

Figure 2 shows a computer organization with memory protection. The three main components are the central processing unit (CPU), the memory management unit (MMU), and the memory. Each component has a counterpart in the reference monitor model. The CPU provides the data manipulation functions, and is analogous to a subject. The CPU transfers data between memory and itself. The memory is conceptually where the objects are located; objects accessible by a subject are grouped together in a domain. Located in the memory are the security-related objects such as the access matrix and the policy manager. The MMU is analogous to the policy enforcement elements of the reference monitor, as it ensures the domain separation and does the runtime verification of object access rights. The MMU uses memory-resident tables to define the physical location of each object within the memory space and the access rights to it. These tables make up the access matrix, and their entries are set up and maintained by the access policy manager in the trusted computing base (TCB).

3. GRANULARITY

The protection mechanisms of a machine can define several distinct objects and types of objects. This distinction is necessary for the machine to maintain process separation. Objects can be described by type, number, and size. Granularity ranges from coarse to fine. Coarse granularity means that objects have the minimum distinguishability necessary to support a system with minimal security. In such a system, there are only a few objects and object types defined by the machine. A machine supporting fine granularity allows the system designer to define minimal sized objects and allow for many objects within a domain. This allows the system to enforce least privilege. To be useful, the definition of an appropriate number of objects must also be supported. A domain is a special object type where access is by transfer of control and can be a part of another domain. As such, the measure of granularity is based on: 1) objects, and 2) collections of objects (or domains).

3.1 Domain Granularity

Domain granularity is a combination of the number of simultaneous domains supported, the control of interdomain transfer, and the distinction of domains. With coarse granularity, transfer is controlled in only one direction between two domains, and the hardware can distinguish between only two domains. With fine granularity, the hardware can simultaneously separate many domains, and control direct transfer between any of the domains. Transfer of control from a domain can be limited to a subset of all domains. The distinction of domains is called domain isolation.

3.1.1 Interdomain Transfer

Interdomain transfer is the transfer of processor control from one domain to another. To do this, a subject executes a machine instruction that passes processor control to another subject that exists in a different domain[2]. An example of interdomain transfer is an application program requesting an operating system service. In this event, the operating system gains control within its own domain. Other aspects of interdomain transfer are controlled entry of the called domain, return back to the calling domain, and parameter passing between the domains.

The machine must ensure that during interdomain transfer, the called domain is entered at a specific location called an entry point. The entry point is where the subject in the called domain starts executing. No subject should be allowed to modify where another domain is entered, or enter at a location of its own choosing. A machine that cannot enforce domain entry points can never prevent subversion by hostile software. Without any entry enforcement, an attacker is able to enter a more privileged domain where the security checks can be bypassed.

The return to the calling domain is not strictly enforced on most machines. The return location for the calling domain is placed either on the processor stack or in a specific location in memory. The called domain is usually not prevented from modifying this information, and is trusted to use the supplied return address. In these instances, the called domain is a more privileged domain. At the finer granularity of domain isolation, the machine also ensures that the calling domain is reentered at a known place. The return information is placed at a location inaccessible to either domain, preventing the return address from manipulation.

Another issue is the information transferred between domains. Information passing provides a service domain with the information needed for its task. The more privileged domain transfers the data, as it can access the source domain's address space. If the data transfer software is not implemented properly, a potential route for attack is opened up. With fine domain granularity, the transfer is done by the hardware, as one domain cannot observe the contents of another.

3.1.2 Isolation of Domains

Domain isolation is the machine's ability to keep any subject from observing the contents of another domain. Ideally, a machine should prevent a subject from observing the objects in another domain. In most machines, only two active domains are supported at the same time. One domain has access to all objects within the system, and is considered the privileged domain. The second domain has access to a subset of the objects, and is called the user or applications domain.

Domain isolation granularity is the number of simultaneous domains a machine can support, and the allowable access privileges to other domains. A large number of domains allows a system developer to separate software across several domains. This capability can improve software safety, because side effects are minimized and program errors can be detected. A subject's domain execute access privileges limits the domains directly reachable by a subject. With this mechanism, functions can be separated into different domains, and access is enforced by hardware rather than by the operating system.

3.1.3 Coarse Domain Isolation Granularity

A machine must provide a mechanism for the TCB to protect itself. In coarse granularity, this protection is provided by giving the TCB a facility to segregate the untrusted software into a single domain, which is a subset of the address space of the machine[9]. The machine has to support two domains: one has access privileges to everything, and the other has restricted privileges[3]. The TCB domain can execute the entire instruction set of the machine, access all devices, and access the entire address space. The second, non-privileged domain, is constrained to a subset of the address space,

[2] At the ISA level, the registers, the tables that define the address space to the MMU, the stack pointer, and the instruction pointer represent the context of the process. The context must be changed to represent the new subject and domain during an interdomain transfer and is known as a context switch.

[3] A two domain machine is analogous to what is commonly known as a "two-state" machine. For consistency, the notion of domains is used.

and cannot execute instructions that can circumvent the hardware protection[4]. These protection relevant instructions are also called privileged instructions. An example of an architecture that provides coarse domain isolation granularity is shown in Figure 3. The least privileged domain is confined by the base and bounds registers. A state bit is used to show that address translation is based on the base and bounds register, and the privileged instruction set cannot be used.

3.1.3.1 Coarse Control of Interdomain Transfer

With coarse granularity, interdomain transfer is controlled only for entry into the privileged (called) domain from the unprivileged (calling) domain. The transfer is done by the subject executing an instruction that causes a domain transfer. The privileged domain entry point is either a single location, or is determined by a vector table with the calling domain providing an index as part of the instruction. The return to the calling domain is under the control of the called domain. The hardware-generated return address is accessible to the privileged domain and can be manipulated. Transfer between unprivileged domains must go through the privileged domain; this degrades performance because two context switches are needed and the operating system must validate the transfer. The aforementioned are problems in secure systems[5].

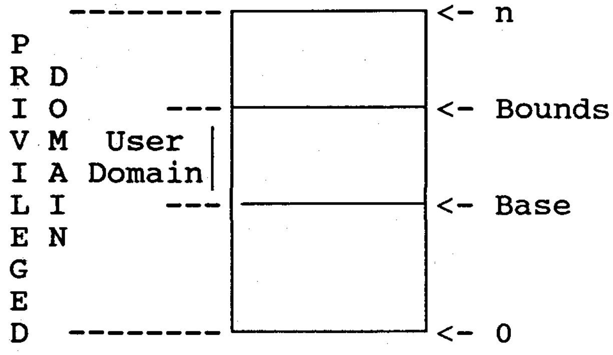


Figure 3

3.1.3.2 Coarse Isolation of Domains

Domain separation is minimal with coarse granularity. The lesser privileged domain is restricted to a subset of the address space. Its physical start location is set in a base register and the length is placed in a bounds register. The lesser privileged domain can observe only a logical space that, to it, starts at zero, and has a maximum address which is the size in the bounds register. It has no idea of its true location in memory. The TCB domain has access to all memory, including that of the lesser privileged domain.

Multiple domains can be simulated by the privileged code changing the appropriate hardware registers over time, to define the additional domains. This manipulation gives the illusion that the machine is supporting multiple domains, while only two domains can exist at a time. A finer domain granularity machine can be simulated, but experience has shown that the kernel is more complex, error prone, and also reduces the overall system performance, as was shown in the early Multics experience[12].

3.1.4 Medium Domain Isolation Granularity

Medium domain granularity is characterized by multiple domains, where each domain of lesser privilege is contained within a domain of higher privilege, giving a hierarchy of domains as shown in Figure 4. Machines with this characteristic are called ring architectures. One domain has privilege to access the entire machine and one domain has minimal privileges. In between, the domains can have increasing privileged access to certain hardware functionality such as I/O devices. These intermediate domains can access only domains of lesser privilege. Processes can use this facility to protect sensitive

[4] Instructions are also considered as objects. There are at least two groupings of instruction objects. The base instruction set which every subject can execute, and the privileged instruction set that contains instructions that manipulate processor control registers and devices. A subject can use the privileged instruction set to circumvent the protection mechanisms. As such, the subjects that have execute access to these instructions must be minimal. In this example, the privileged domain has access to an additional object, known as the privileged instruction set.

data from exposure to other parts of the process, and to implement a runtime that needs additional privileges.

3.1.4.1 Medium Control of Interdomain Transfer

Interdomain transfer is done directly from one domain to another within the hierarchy. Since only one process is represented at a time, transfers to other processes must go through the most privileged domain for validation and manipulation of the control registers. The calling domain has no knowledge where within the hierarchy the transfer is destined. The called domain must be of either equal or greater privilege than the source domain. The calling domain cannot manipulate the pointer to the entry of the domain being called. The return point can be manipulated by the called domain. This is because the return address is placed upon the called domain's stack; however, a return is restricted to a domain of lesser or equal privilege. This rule prevents the use of the return mechanism to attempt surreptitious entry into a more privileged domain.

3.1.4.2 Medium Isolation of Domains

The domains of greater hierarchical privilege have access to the objects of lesser privileged domains. The most privileged domain can access all of the objects of the system. This domain structure allows a process to have multiple levels of privilege. This model assumes that the privileges desired within a process are hierarchical. The more privileged domains can be allowed access to I/O devices while the least privileged domain is denied access. Device manipulation can take place without the device driver being in the most privileged domain.

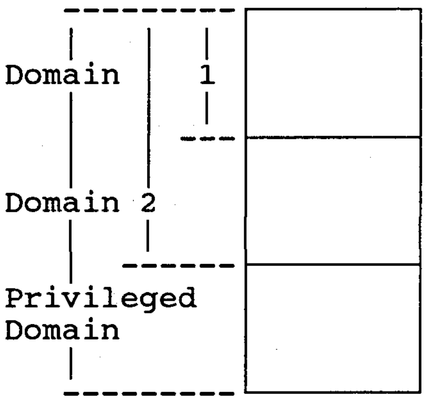


Figure 4

3.1.5 Fine Domain Isolation Granularity

Fine domain granularity is characterized by multiple non-hierarchical domains, where each domain can exist as an independent address space. This allows for small subdomains to exist within a program, such that each subprogram has the minimal data for its needs[10], supporting least privilege. Subdomains ensure that procedures do not affect the data of other procedures within the same process, by enforcing data hiding and minimizing side effects. This machine can support a system such that the most privileged subject can observe only its own domain. As a result, there need not be a supervisor domain or state on this type of machine.

3.1.5.1 Fine Control of Interdomain Transfer

For fine domain granularity, transfer between domains is strictly controlled for both call and return since the destination domain cannot manipulate the return address. A separate control area is used to hold the return location and the saved context. This area is located in another domain inaccessible to both domains. An additional requirement is that the destination domains allowed for a subject are restricted by the hardware. Each subject has an associated list of domains that can be called which is maintained by the operating system. The hardware enforces this list, and the software need only check when a new domain is added to the list. With this feature, a service domain can be directly called without having to go through the operating system, providing a performance benefit. In addition, the services a subject can directly utilize can be limited by having the operating system set the allowable domains at process initialization. Access to a service can be deleted at any time by the operating system.

3.1.5.2 Fine Isolation of Domains

The machine can isolate every domain within the system at fine domain granularity. No subject will be able to access the objects of any other domain. Least privilege can be fully enforced on this system. There is no single domain that can view every object within the system. In a machine such as this, the system can be designed such that the TCB's actions are constrained[5]. Objects, such as instructions and access to I/O hardware, can be assigned on a per domain basis. This provides the system developer assurance that certain functions are carried out in specific domains only, and not anywhere else within the operating system. Malicious code is difficult to inject into in an operating system that adheres to strict least privilege. With a coarser granularity, injection is necessary in the TCB domain only. With fine granularity, proper system design would force the attacker to inject code into more than one domain, because of the enforced separation of privileges. The same approach can be taken in the design of applications software.

3.1.5.3 Discussion

Fine granularity machines require built-in operating system features. Facilities, such as context switching and process dispatching, have to be done in hardware; otherwise a supervisor state must exist to change the processor context. When operating system assists are part of the architecture, one could argue that the supervisor domain is in control when the microcode that implements those instructions is executing.

Interprocess communication (IPC) has to be done in hardware; otherwise, an intermediate domain must exist to move data between two domains. Some machines support message passing in hardware. This facility would reduce the amount of overhead in secure systems. Many operating system services are written as trusted subjects and a great deal of system overhead is involved in passing information from the requestor to the service because of the amount of validation and context switching involved.[5] The ability to call or send a message directly to a service significantly reduces the overhead involved.

The number of domains to use is a tradeoff between performance and security. A large number of domains can result in a relatively small number of instructions executed within each domain which can degrade performance. Domain transfer requires a context switch which is expensive because the CPU state must be saved and replaced by another. This state information can be extensive, and the exchange can consume a lot of machine cycles. The cost of a hardware domain switch, however, is less than the corresponding software implementation.

3.1.5.4 Advantages of Fine Granularity

Fine granularity systems can provide both a performance and a security benefit. In particular, less operating system code is necessary, as certain primitive operations are implemented as a part of the machine. These functions include process dispatching, interprocess communications, and direct domain transfer. The operating system can be a set of servers each confined to its own domain. Additional advantages include greater reliability in the construction of programs, because small protection domains can catch software anomalous behavior resulting in fewer side effects.[10]. In addition, the effects of malevolent code are also constrained by the strict enforcement of least privilege, so that the propagator must subvert more than one module, making the task more difficult.

3.1.5.5 Disadvantages of Fine Granularity

There is a significant cost, associated with increased hardware complexity, for implementing fine granularity features. Correctness is an issue, as an implementation could be conceptually correct but poorly implemented thus creating a security hole. The proper design of a fine granularity system is also an important consideration. For example, capability based systems[2] fall in the class of fine granularity protection. There are certain classes of capability systems that are inadequate for supporting a mandatory security policy[8].

3.2 Object Granularity

Object granularity is the ability of a machine to represent a number of different objects and object types within a domain. Different object types are the process address

[5] However, there has to be at least one domain that can manipulate the tables that control the MMU. This domain is analogous to the access matrix monitor. With fine granularity domain isolation, we can ensure that only the code in this domain can manipulate these tables, and this code is short. On a static system, the MMU tables can be preset, and no access allowed to them at all.

space, the privileged instruction set, the control registers, and other domains. At coarse object granularity, a nonprivileged domain has to have at least two objects, representing two different object types, for the system to provide protection. One object is the process address space, and is a memory object. The other object is a link to a privileged domain, implied by an instruction that calls that domain. In a machine with fine object granularity, there can be multiple objects and object types within a domain. The remainder of this section assumes objects to be memory objects, though much of the discussion can be applied to the other object types as well.

A memory object is a linear address space called a region[6]. A subject has a limited set of privileges to each region. Typical privileges include: read, write, and execute. Privileges can be set on a page or subregion on some machines. For example, an object can be defined as a contiguous group of pages with the appropriate privileges set. The page size defines the minimum size of an object, though the size of the object is actually smaller. Some systems allow for variable page sizes and consequently fit the object size better. The ability to set privileges further subdivides the object granularities.

3.2.1 Coarse Object Granularity

A single region or single linear address space characterizes one form of coarse object granularity. Objects mapped into a linear address space are indistinct, unless the object privileges can be set over the subset of the region. When privileges are set on subregions, the subject still can inadvertently manipulate an object, because there are no bounds on addressing within the domain.

3.2.2 Fine Object Granularity

Fine object granularity is characterized by multiple regions or linear address spaces within a domain. With the proper privileges and number of regions supported, each object can be represented by a region. The subject uses an additional addressing parameter called a descriptor, to specify the objects on which to operate. This selection minimizes the chance for an object to be modified by an instruction side effect. The TCB can decide the object privileges a subject can have, and a domain can have as many objects as there are address spaces.

3.2.3 Observations

The maximum object address space is important. During the development of the Secure Communications Processor, it was felt that a small segment size would be important in aiding the enforcement of least privilege[3]. A somewhat larger segment size of 64K was a part of the Intel 808X/80286 architecture. In both cases, the address size had a direct impact on performance and the ability for the processor to support large arrays. Large arrays require multiple segments and additional software complexity to support indexing.

3.3 Operating System Assists

Operating system assists are machine provided functionality that can reduce the amount of operating system code and are intended to improve overall system performance. They are related to security because of the added complexity to the machine and, if implemented improperly, can provide an avenue of attack for a penetrator. In addition, operating system assists can be implemented such that certain operating system functions can be restricted to a specific domain, thus supporting the principal of least privilege. In machines that support fine granularity, these functions tend to be placed in the hardware.

3.3.1 Non-mastermode Fielding of Interrupts

On most architectures, interrupts are handled in the most privileged domain of the system. This approach is taken because the interrupt handler must be able to access the

[6] The term region is used here instead of segment to reduce confusion. In practice, the term segment has two meanings. The first meaning relates to a level in the page table hierarchy like the Motorola 68000 series and the IBM System 370. In this context, a segment is a grouping of pages that is within a linear address space. Data are accessed on the basis of the linear address, and access is limited based on the permissions and the existence of a page in the virtual address space. The second definition of segment is multiple linear address spaces within the domain. Here the subject decides which of these address spaces to operate on. To access data, the subject must indicate which address space, and provide an offset within that address space.

device that posted the interrupt; however, the interrupt handler does not need the full privileges of the system. The interrupt handler needs only those privileges that allow it access to the device in question. This facility allows for a device to be controlled from a separate domain.

3.3.2 Process Dispatching

When the control of the machine is changed from one process to another, the processor context must be changed. This operation means that the executing software, called the dispatcher, has privileges to the entire machine, since the state of the machine must be changed. This operation is similar to the fine granularity domain change. Isolating this operation to an instruction that does a context switch would allow the system to constrain the dispatch software to a domain, limiting its access to other areas of the system. Optimally, the dispatch code can be implemented as a part of the machine. The operating system would then only need to do task scheduling.

3.3.3 Parameter Passing as part of Interdomain Transfer

When one domain transfers control to another domain, it is usually done for the purpose of requesting a service. The called domain needs information from the caller which is provided as parameters. The called domain must move the parameters into its space to prevent the calling domain from changing the parameters in an attempt to bypass checks. The operating system should handle this transfer to prevent this type of attack and to prevent the called domain from viewing the calling domain.

3.3.4 Restricting the Actions of Input Output/Devices

Restricting the actions of input and output (I/O) devices is a system function rather than a processor function. It is discussed in this context because there is usually no control over how data moves between secondary and primary storage. I/O devices normally have access to the total memory space, even though they normally act within the confines of an unprivileged process. Restricting the activities of these devices can be done by using the same tables that restrict the activities of the processor. The overall security of the system is improved because the device can view only the space where data are to be transferred, which makes user-controlled devices more feasible. In addition, the device drivers are relieved of the need to do the virtual to real addresss translation which reduces the amount of privilege needed for them. This provision could be implemented either by having a MMU restrict the actions of the I/O devices[11] or by having the CPU move the data from the I/O port to the specific address under CPU control[6]. Without bounding the actions of I/O devices, extensive checks must be done on the command sequences to ensure that data transfer stays within bounds[12].

3.3.5 Interprocess Communication

Interprocess communication (IPC) is a facility provided in the operating system that allows two different processes to send data to one another, and to synchronize their actions. This activity requires the operating system to access two different domains at the same time, and to determine if these domains can communicate.

4. SUMMARY

The processor features necessary to support a trusted system have been discussed in this paper. Processor protection mechanisms confine subject access to other domains and to their own objects. Figure 5 maps instruction set architectures (ISA) onto a grid formed by the domain and object constraints. This map can be used for a relative comparison of ISAs, and can show that a processor can be coarse for domain granularity but fine for object granularity. For illustration purposes, several commercial processors are placed in the Figure. This mapping cannot be exact for each processor, but one can get a relative idea of how the various processors compare. The most desirable architectures are those that are located in the lower right hand corner of the figure. At this location, the machine can best enforce least privilege for both domain and object access. Along with the necessary operating system support, overall system performance is also improved. The design and implementation of a machine that meets the criteria of fine object and domain granularities must be carefully considered. Machines that are categorized here are relatively complex, with the associated implementation problems.

		DOMAIN GRANULARITY		
		Coarse	Medium	Fine
OBJECT GRANULARITY	Coarse	68030	VAX 11/780	
	Fine		80x86	80960XA

Figure 5

5. REFERENCES

1 J.P. Anderson, *Computer Security Technology Planning Study*, ESD-TR-73-51, vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, Bedford, Mass., Oct. 1972.

2 R.S. Fabry, "Capability-Based Addressing," *Communications of the ACM*, vol. 17, no. 7, pp. 403-412, 1974.

3 L.J. Fraim, "SCOMP: A Solution to the MLS Problem," *IEEE Computer*, vol. 16, no. 7, 26-46, 1984.

4 G.S. Graham and P.J. Denning, "Protection-Principles and Practice," *AFIPS Spring Joint Computer Conference*, pp. 417-429, 1972.

5 B.D. Gold, R.R. Linde, and P.F. Cudney, "KVM/370 in Retrospect," *Proceedings of the 1984 Symposium on Security and Privacy*, pp. 13-23, 1984.

6 *IBM Technical Reference-Personal Computer AT*, S229-9611, IBM Corporation.

7 B.W. Lampson, "Protection," *Proceedings of the 5th Annual Princeton Conference*, Princeton University, pp. 427-433, March 1971.

8 R.Y. Kain and C.E. Landwehr, "On Access Checking in Capability-Based Systems," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 202-207, 1987.

9 C.E. Landwehr, "Hardware Requirements for Secure Computer Systems: A Framework," *Proceedings IEEE Symposium on Security and Privacy*, pp. 34-40, 1984.

10 T.A. Linden, "Operating System Structures to Support Security and Reliable Software," *Computing Surveys*, vol. 8, no. 4, December 1976.

11 S. Namba, N. Ohno, H. Kubo, H. Morisue, T. Ohshima, and H. Yamagishi, "VM/4: ACOS-4 Virtual Machine Architecture," *Proceedings 12th International Symposium on Computer Architecture*, pp. 171-178, 1985.

12 G.J. Popek and C.S. Kline, "Issues in Kernel Design," *Proceedings of the National Computer Conference*, pp. 1079-1086, 1978.

13 R.F. Rashid, "Threads of a New System," *Unix Review*, Vol. 4, No. 8, pp. 38-49, 1986.

CONDUCTING AN OBJECT REUSE STUDY

David R. Wichers

Arca Systems, Inc.
2841 Junction Avenue, Suite 201
San Jose, CA 95134

Abstract[1]

This paper describes the Object Reuse requirements defined by the National Computer Security Center's (NCSC's) Trusted Computer System Evaluation Criteria (TCSEC) [2]. It explains what object reuse is, why object reuse is a concern for trusted systems, presents a methodology for conducting an object reuse study that will determine whether a system satisfies the TCSEC requirements for Object Reuse and then discusses some other issues not covered by the methodology.

Introduction

The National Computer Security Center (NCSC) was established in 1981 to address the need for trusted systems within the federal government. The NCSC has developed one standard and numerous guidelines for the development of trusted systems. The backbone of these is the Trusted Computer System Evaluation Criteria (TCSEC) [2] which defines the requirements that systems must meet to achieve specified levels of trust. One of the areas of concern indicated by the TCSEC is the reuse of any storage medium that contains an object which was deleted. This paper explains why Object Reuse is a concern for trusted systems, discusses various issues in this area and describes how to conduct an object reuse study that will determine whether a system satisfies the TCSEC requirements for Object Reuse.

What is Object Reuse?

"Object Reuse" (or "Allocation Residuals") refers to the problem of information left behind on media used by a previous object that is reused to form another object. When an object is discarded, the medium (memory, disk, tape, etc.) on which it is stored still contains information that must be protected according to the rules of the system security policy. As such, a system must ensure that this information cannot be exposed in an unauthorized manner. When an object is formed from system resources, care must be taken to ensure that all previous information has been erased and all previous rights to those resources have been removed before the resources are reused. For example, suppose a file that contains Top Secret information is deleted. When this file's disk space is reused, the operating system must ensure that all of the data in this disk space is overwritten before it is allocated to another file. Before examining object reuse in more detail, we need to understand exactly what an object is.

The TCSEC defines an object as:

> "A passive entity that contains or receives information. Access to an object potentially implies access to the information it contains. Examples of objects are: records, blocks, pages, segments, files, directories, directory trees, and programs as well as bits, bytes, words, fields, processors, video displays, keyboards, clocks, printers, network nodes, etc." [2] (Glossary pg 114)

[1] This research extended work previously funded by by the National Computer Security Center under contract number MDA904-89-C-6022.

In a C2 system, where a large portion of the operating system is trusted, the typical items that are thought of as objects by a user are those entities explicitly recognized by operating system calls. These would be files, directories, processes, messages etc. Access to these objects is usually controlled directly through operating system calls. A user does not usually think of records, blocks, pages, segments, bits, bytes, or words as objects. These entities are the physical building blocks that an operating system uses to create system objects. In more highly trusted systems, these more primitive objects (especially segments) may be the objects directly protected by the Trusted Computing Base (TCB). Object reuse is concerned with the protection of the contents of these physical building blocks when the object that was using them is discarded.

The TCSEC defines object reuse as:

> "The reassignment to some subject of a medium (e.g., page frame, disk sector, magnetic tape) that contained one or more objects. To be securely reassigned, such media must contain no residual data from the previously contained object(s)." [2] (Glossary pg 114)

For TCSEC assurance class C2 and above, the following is required to ensure that object reuse is accomplished in an appropriate manner. The TCSEC object reuse requirements do not change beyond C2. These requirements are:

> 1) *All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects.*
>
> 2) *No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.*

The TCSEC definition of object reuse states that "no residual data from the previously contained object" can still exist on the medium when it is reused. There is another concern that must be addressed by object reuse besides the requirement to purge residual data before it is reassigned. This additional requirement is indicated by the first TCSEC requirement which states that all authorizations to an object must be revoked as well. As such, object reuse is concerned with the exposure of discarded information as well as the removal of authorizations to the discarded containers. Examples of both types of object reuse errors follow:

Content Residual Errors:

1) A disk block is assigned to a new file without being purged or overwritten.
2) Stack space with stack information from a previous process is assigned to a new process.
3) A newly assigned file descriptor contains the date of last modification from the previous file to which the descriptor was assigned.
4) Data at the end of a scratch tape is not overwritten before the tape is assigned to a new process.

Access Residual Errors:

5) A file is deleted that has multiple entries in the file system. The disk space is returned to the free pool, yet other file system entries still refer to the freed disk space.
6) A process spawns a child process that has access to the parent's data space and then the parent process terminates. The parent's data space is returned to the free pool without removing the child's access permission to the freed data space.
7) A file is deleted while a process has the file open. The disk space is returned to the free list without removing the process's open file descriptor to the deleted file's disk space.
8) A system that supports semaphores does not revoke all rights to the semaphore when it is deleted by the owner process. When the semaphore is reused, a previous process still has access rights to the new semaphore.

Object Implementation Issues

Having defined object reuse we now need to understand how objects are implemented by a system. We need to know where objects reside, how they are initially created, allocated and finally deallocated.

When storage media is initialized by the system, whether it be a new disk, new tape, or the system memory on powerup, various portions are dedicated for specific uses, but the bulk is divided up and placed in a free pool or pools for later use. These free pools contain the building blocks (components) that are used to make new system objects. These blocks are the primitive units of the media that are protected and used to store information. When an object is created, components from a free pool are allocated to the object. When an object is deleted, its components are returned to their respective free pools. These initialization, allocation and deallocation mechanisms must be examined for each type of component to ensure that proper object reuse procedures are carried out correctly.

The combination of initialization/allocation or deallocation/allocation processing must ensure that all access and content residuals are removed before access is granted to the next process. The exact responsibilities assigned to the initialization, allocation, or deallocation mechanisms is up to the system designer, who must weigh such factors as security, functionality, efficiency, user response time, etc., when making such decisions. Regardless of the reasons for the implementation decisions, the combination of these mechanisms must ensure that all data is erased before an object is reused and all access rights to an object are revoked when the object is placed in the free pool. Below we describe in more detail what must be done by the deallocation, initialization, and allocation mechanisms.

Deallocation

Deallocation processing involves the deallocation of an object that has been deleted or is no longer in use. Care must be taken when access to an object is removed that all current accesses to the object are gone before it is deallocated. The deallocation mechanism must revoke access to the object that was just released and then perform any content residual operations necessary for the object type. This could include partial (to be completed during allocation) or complete purge of the information contained in the object. Once deallocation processing has been performed, the individual pieces of the object can be placed in their respective free pools and marked available for reuse.

The concept of revocation is simple but some of the issues that arise because of it are not. Even though there is usually only one instantiation of each object, there are often multiple references to an object and these may be of different types. For example, a file is stored on some type of medium and is referenced through a file system entry. Multiple file system entries may exist for the same file. In this case there is only one physical copy of the file but many references to it. When a process deletes this file it is actually deleting a file system reference. If the process deletes the last file system reference then the actual file is deleted as well, assuming there are no current accesses to the file.

This brings up another issue that complicates revocation. In addition to file system references, processes may have files open for reading or writing (have current access) as well. If a file's last file system reference is removed then the file is to be deleted. If there are no current accesses to the file it is deleted at this time. If a current access to the file exists then one of two things may happen. Depending on the system's security policy this deletion may cause all current accesses to be immediately revoked and the object deleted, or cause the file to be tagged for deletion after all the current accesses are removed voluntarily.

Other objects may be similarly complicated. Multiple processes can be executing off of the same executable image. When one of these processes exits, the executable image must not be deallocated while the other processes are still using the image. Processes can also share memory. When a process exits, its current access to any shared memory is revoked but the shared memory itself is not deallocated until all current accesses are removed.

As one can see, the revocation processing may be complicated by a number of different scenarios. The actual deallocation processing can be triggered by many types of events. Each of these must be examined to insure that they invoke the proper revocation mechanism and the mechanism performs the correct revocation processing.

Initialization

Initialization processing involves the creation of the object free pools and the possible division of the available resource into units for later use by newly created objects. If the allocation mechanism for a resource depends on the deallocation mechanism to carry out certain operations, equivalent operations must be performed by the initialization mechanism. For example, if an allocation manager for heap memory depends on the deallocation mechanism to purge the contents of the heap memory before it is placed back in the free pool, the initialization procedure must purge the contents of all the heap memory before placing it in the newly created free pool. Otherwise, the previous contents of the heap could be exposed to the first process that is allocated any unpurged heap space. The initialization mechanism could also mark the heap space as unpurged, which would then require that the allocation mechanism make a special check for this flag and purge the space itself if needed.

The object reuse study must verify that the combination of the deallocation and allocation mechanisms and the combination of the initialization and allocation mechanisms perform all the necessary access and content residual processing. A clean implementation would have the free pool initializer leave the pool in such a state that no special processing would be required when an initialized object is allocated for the first time. The major concern with regards to initialization revolves around content residuals, not access residuals. Since access rights to an object are defined during the object's initialization, it should be trivial to ensure that a newly initialized resource does not have any access residuals.

Allocation

Allocation processing involves the allocation of a component from a free pool to a newly created or expanded object. This mechanism must complete the object reuse processing started by the initialization or deallocation mechanism. It must ensure that all access and content residuals have been revoked prior to granting access to the requesting process. If the contents of a component have not been purged before this time, it must be performed by the allocation mechanism.

Purging can be done in three different ways. The data can be overwritten with a fixed pattern, a random pattern, or it can be overwritten with data provided by the user before the user is granted read access to the object. This last mechanism is referred to as 'write before read' meaning that the user must overwrite the space used by the new object before the user can read it. This mechanism is typically used for efficiency reasons to avoid the necessity to purge the space explicitly. However, what the user writes does not typically match the exact size of the allocated container. This unpurged space at the end must either be overwritten with a fixed or random pattern or the user must be restricted from accessing the space until it has been overwritten with user data. An example would be disk space allocated for a file. Typical end of file processing by an operating system would prevent a user from reading beyond the end of the user's file, thus restricting the user to data generated by the user and protecting any residuals that lie at the end of the file's last disk block.

The allocation mechanism must also initialize all attributes of the object that are visible to the user. For example, when creating a file, a file descriptor can contain a wealth of information about the newly created file, such as file size, date and time of creation, date and time of last read, date and time of last modification, discretionary and (possibly) mandatory access control information, filename, filetype, etc. Any information of this nature that could possibly be carried over from a previous use of the object must be purged or overwritten. Once all residual processing has been completed, the object components must be removed from their free pools, marked as in-use and an access path created from the process to the object.

Free Pool Protection

The previous discussion has revolved around taking objects from free pools and placing them back in free pools. Objects currently residing in a free pool may contain information that must be protected if it was not purged by the initialization or deallocation mechanism. To ensure that this information is not exposed in an unauthorized manner, each free pool must be protected from unauthorized access.

Typically, all the free pools that reside on the same physical medium will be protected by the same mechanism. For example, a system's main memory will typically contain a number of different free pools, such as disk buffers, tape buffers, message buffers, user stack, data, execution and heap space, etc. Each of these components will usually have its own free pool rather than a system wide free pool of main memory that any of these component types can use as needed. Some of these components may share a common free pool, such as stack, data, and execution space, but it is unlikely that all components that reside in main memory will share the same common memory pool. The process isolation mechanism for the operating system, such as a memory manager, will, in addition to providing process isolation, typically protect the operating system from user processes as well as protect all the free pools that reside in main memory. This single mechanism may protect the majority of free pools that reside on a system. Each mass storage device (disk, tape, etc.) that contains a free pool should be protected by whatever mechanism is normally used to prevent unauthorized access to devices. Free pool protection mechanisms should be an inherent part of the operating system protection mechanisms, not an additional feature specifically addressing object reuse requirements. The protection mechanism for each object type should at least be noted for completeness in the study of the object reuse mechanism.

Methodology

This section presents a methodology for an object reuse study that will determine whether a trusted system meets the TCSEC requirements for object reuse. The steps in the methodology are listed below and then examined in the following section.

1) Identify all system objects as described by the system's security policy model (if it exists) and the system documentation.

2) For each system object, determine the object components the system uses to create the object.

3) From the previous step, identify all the components supported by the system.

4) For each of these components, identify where they are initially created by the system, when they are allocated to system objects and when they are deallocated and returned to their free pools.

5) Each component's initialization mechanism must be examined to verify that the proper initialization is performed to ensure that no access rights exist to the newly initialized components and all previous information will be purged when the component is allocated.

6) For each component, verify that a proper purge or overwrite is performed for each object placed on and removed from a free pool during deallocation and allocation.

7) For each component, verify that all previous rights to an object in a free pool are revoked prior to reallocation.

8) For each system object, identify all operations that can be used to allocate or deallocate the object. For each of these calls, ensure that the mechanisms examined above in steps four through seven are used to allocate and deallocate the components that compose the system

object. If not, then each exception must be examined to ensure that they enforce the requirements checked in steps six and seven.

9) For each free pool, the mechanism used to protect it must be identified and examined to ensure that any unpurged information in a free pool is protected.

An explanation of these steps follows:

Identification of System Objects

First we must develop a comprehensive list of all object types protected by the TCB. If there is a system security policy model (required at TCSEC levels B1 and above), this list can be derived from the model. Otherwise, if a security policy model does not exist for the system, it should still be a straightforward task to identify all the object types controlled by the system. The external documentation of the system should clearly identify all of these objects. Identification of each type of medium supported by the system may be beneficial in this search as well [3]. Every type of storage device will contain some form of an object. Not only does each object type need to be identified, but it must also be determined where, and in what form, any portion of each object may reside (Disk, Tape, Removable Disk, RAM Disk, Cache, etc.) During the study, each device driver may need to be examined to ensure that proper object reuse processing is performed for the objects supported by the device if the object reuse mechanism relies on any special calls provided by the device driver.

Identification of Components

The next step requires considerable knowledge of the implementation details of the operating system. The composition of each of the identified system objects must be determined. For example, a file, in all of its various forms, is composed of a number of different system resources. Files usually have a directory entry (possibly multiple) that provides a link to the file's descriptor. The file descriptor keeps track of information about the file, such as name, owner, size, dates of creation, access rights, modification times, file's physical location, who has it open, etc. The file itself must reside on some physical medium (tape, disk, etc.) which means it is composed of a number of blocks of the medium plus information describing where these blocks are located on the device. If the file is being accessed by a process then portions of the file will reside in main memory as well. So when one talks about use or reuse of a file, one single entity does not represent the entire file.

We call the pieces of a system object components. Virtually all objects are composed of at least two components, a header component describing the object and the object itself. In the file example above, it was shown that files are made of many different components. The list would include: a directory entry, file descriptor, portions of some physical medium, an index on the medium describing exactly where these portions are, blocks of main memory that contain portions of the file, an index describing where these portions reside, portions of the device cache, etc. A simple object such as a semaphore could be composed solely of a descriptor and a single bit or word for the semaphore itself.

For each of the system objects identified in the previous step, each of its components must be identified. Once this is done, the entire set of components for the system can be determined. Different object types can share a common component. In UNIX[2] for example, files and directories are two different system objects, but their internal composition is identical. They are both managed by the file system manager and composed of the same components [1]. By combining all the components found in this exercise and eliminating redundant ones, the entire set of system components can be determined.

[2] UNIX is a registered trademark of AT&T.

This set of components describes the actual "objects" that are subject to object reuse and the subject of the bulk of the analysis effort described below. At this point one finally knows what the actual subject of the study is. In order to show that the identified system objects are properly reused, it is both necessary and sufficient to show that each of the component managers appropriately implements the TCSEC object reuse requirements listed above. Reuse of each of these components must be examined to ensure that all data is erased before they are reused and all allocation residuals are removed.

Verification of Proper Object Reuse Processing

This is where the study of the actual object reuse processing begins. For each component type three things need to be identified:

1) Where they are initially created by the system.
2) Where they are allocated to system objects.
3) Where they are deallocated and returned to the free pool.

Once identified, the object reuse processing performed by each mechanism needs to be examined. For each component, it must be verified that the object reuse processing performed by the initialization/allocation mechanism is equivalent to that performed by the deallocation/allocation mechanism or that a complete and proper initialization is performed to ensure that all previous information is purged and no access rights to the newly initialized components exist.

Given the above, it remains to verify that the deallocation and allocation mechanisms perform the proper object reuse processing. For each component, it must be verified that a proper purge or overwrite is performed by the combination of the deallocation and allocation processing. Special attention must be taken to ensure that all possible header information is overwritten as well. For each deallocated component, it must be verified that all previous access rights are revoked prior to reallocation. It is essential that a user who had access rights to a discarded object not be able to access the associated space after it is in the free pool and later reallocated to a new object.

Tracing the System Call Mechanisms

The previous step examined the allocation/deallocation mechanism for each component type. The next step is to verify that all the system calls that can be used to create or delete an object invoke the mechanisms that were previously analyzed. This analysis is facilitated by the use of small, simple resource managers which demonstrably provide the sole means of accessing that component type. If each system call that accesses a component uses a resource manager call to manipulate the component, then only the resource manager has to be analyzed. For each system call, the calling chain must be traced down to the point where it actually invokes one of the previously examined resource manager routines to allocate or deallocate each component of the system object. If it is found that some special allocation or deallocation processing is done outside of these previously examined mechanisms, then this code needs to be examined to ensure that proper object reuse processing is performed.

Any special allocation or deallocation processing done outside of the component manager routines should be viewed with some suspicion. A modular implementation would place all allocation/ deallocation code within each component manager and require that all system call processing invoke these managers when creating or deleting objects. If allocation/deallocation code is duplicated within the operating system, each duplication must be checked and verified to ensure that correct object reuse processing is performed. Placing all relevant code within component managers streamlines the effort required to develop these routines and later verify that correct object reuse processing is performed.

Verification of Free Pool Protection Mechanisms

The analysis up to this point should have determined that proper object reuse processing is performed for each component type. The next step is to verify for each component that its free pool is protected from processes outside the TCB. Free pools need to be protected to ensure that any data contained in an object in the pool cannot be accessed by an unauthorized process. For each free pool, the mechanism used to protect it must be identified and examined to ensure that any information in the free pool is protected. If all information is purged before a component is placed on the free list, the mechanism protecting that free list component need not be examined as closely as a mechanism protecting a free list containing unpurged information.

We refer to this as the "Chicken Coop Problem" because of the analogy to the fox stealing the chickens from the unprotected coop. In general, this problem represents a significant system vulnerability and may be the target of concerted penetration testing, which is required at B2. It is most easily eliminated by purging object contents upon release as opposed to at reallocation time. Unfortunately this solution greatly impacts system performance, as much time is devoted to unnecessary purging of object contents that are not reused. In addition, this purge must often occur at awkward times which may impact user response.

Other Issues

An object reuse methodology has now been described. What follows is a discussion of other issues that may be of some concern in this area. A number of these issues deal with the storage of information in places that may not be explicitly recognized as objects, at least at lower assurance levels. Although the explicit object reuse requirements do not change after C2, the additional (architecture, labeling, device, etc.) requirements for B2 and above require that extra attention be made to ensure the protection of information in these other resources.

Identification of these types of resources can be accomplished by enhancing the third step in the methodology. This step currently identifies all the components supported by the system based on the results of step two. An additional requirement for this step at assurance levels B2 and above would be the identification of additional resources that can be reused but are not explicitly recognized as object components at lower assurance levels. This additional step would generate a more complete list of components to be studied for object reuse compliance.

Reuse of Devices

The methodology just described identified and analyzed each resource that could be reused to form a system object. Other reusable resources exist on a system that may not be identified by the methodology. Certain devices are one such example. Each device on the system must be checked to ensure that no device state variables can be used to transfer information between two processes in a manner that would violate the system security policy. This can occur when a device that has been used previously is used again by another process. This reuse of the device makes it a possible issue for an object reuse study. This may be thought of as a covert channel problem and therefore not an issue that needs to be addressed by object reuse. However, although covert channel analysis is required B2, devices are also objects at B2, which then makes this an object reuse problem.

A prime example of this type of device would be a terminal that can retain information from a previous session. Smart terminals can support multiple screens, smart keys, keyboard and screen buffers, answerback messages, etc. These features supported by a smart terminal can accidentally or deliberately be used to "remember" information from a previous session. This information could then be read back by the next process to use the terminal. If the system intends to support a terminal of this type, special logout processing may be necessary to "instruct" the terminal to purge the information retained by these features once the session is completed. Any device that can be directly manipulated by a user process must undergo scrutiny for possible residuals of this type.

Process Registers

Another reusable resource that may not be identified by the methodology are process registers (processor states). When a process is swapped out, all of the processor registers are saved to retain the process' current context. When the process is swapped back in to continue its execution, its context is restored by restoring all the process registers back to the state they were in when the process was swapped out. If some portion of the processor state is not restored during the context switch, then the process may learn something about the previous process, which would be a content residual error. Errors of this kind should be extremely rare as proper context switching must be done correctly by all operating systems to insure correct process execution. An error in the context switching mechanism could cause processes to execute incorrectly, which would be a very serious error to system designers, much more so than any possible object reuse problems.

A more likely sort of object reuse error in this area would be the failure to purge the processor state when a process is initially created. If all processor registers are not scrubbed when a process is first created, then residual information could leak to the newly created process from a previous process. Another area of possible errors would be the failure to restore or purge all processor states of coprocessors and other IC's that can be accessed by the main processor. If a math coprocessor existed on the system for example, and its state was not scrubbed when a new process was created, the state of the coprocessor could be examined by the new process, which would be a content residual error. Residual errors during process creation are more likely than an error during a context switch because initialization errors would not necessarily cause the process to execute incorrectly. However, errors of this type should be very rare as well.

Caches

All of the previous discussion has indicated that all information contained in a component returned to a free pool must be purged before being pulled back from the free pool. This is not always true however. Free pools can be caches as well as irrevocable deallocation pools. If a free pool is a cache, then a later request for something already contained in the cache can pull something out of the cache (free pool) that still contains old information. This is permissible provided the process has the appropriate access rights to the information that was in the object before it was returned to the free pool.

When current access to an object is removed and its components are returned to a cache, these components are not deallocated from the object but rather marked as "available" for deallocation. The cache mechanism must then track the contents of the cache and the association between each component and their respective objects. If portions of the cache are needed for new objects, they are deallocated from their objects and normal allocation processing is then performed. If access to an object contained in the cache is granted to a process, the allocation mechanism has to reassemble the proper components to reform the object from the components contained in the cache. This reallocation mechanism would have to be examined during the object reuse study to ensure that an object is reassembled properly and only information from the same object is returned. Examples of various caches would be Disk or Tape File Caches, Memory Buffer Caches, etc.

Virtual Memory

The virtual memory manager for most systems swaps pages in and out of memory all the time. This mechanism would not usually show up as one of the component managers found during an object reuse study following the methodology described above, unless virtual memory is considered an object, which should be true at B2 and above. The virtual memory manager swaps out chunks of real memory without regard to what object, objects or parts of objects are contained in those pages. This mechanism must be examined by the object reuse study to ensure that the proper pages are restored when they are accessed but have been swapped out. It must also verify that the swap device or devices are protected from unauthorized access by untrusted processes.

Crash Resistant Allocation/Deallocation Processing

An issue that is not explicitly brought up until B3 is that of system crashes and the idea of Trusted Recovery. Object reuse errors can occur if a system crash occurs during allocation or deallocation processing that is not performed in the correct sequence of steps. When developing or analyzing the allocation or deallocation mechanisms, one should keep in mind the consequences of a system crash occurring during any point of their execution. They can easily be developed to be highly resistant to object reuse errors caused by system crashes.

Let's look at the steps typically performed during deallocation processing. The deallocated components need to be placed on their free lists, their contents may need to be purged, and all access rights to the object need to be revoked. If the steps were performed in the order just listed, a crash after their placement on the free list would leave the components unpurged as well as leaving the access rights to these components unrevoked. A safer ordering would be to first revoke the access rights to the object, perform any necessary purges, and finally, placing the object components on their respective free lists. A crash anywhere during this sequence should not cause a residual error to occur. With this approach, the worst thing that could happen would be that the object is left in a limbo state where it is neither owned by a user nor is it on a free list. Standard garbage collection processing would have to find these objects at some point and complete their deallocation processing.

The allocation mechanism can be made similarly crash resistant. The normal steps would be to remove the components from their free lists, complete any necessary purge or overwrite, and then grant access to the object to the creating user. If these steps are performed out of order, a crash could cause the user to have access to the object before it has been purged or overwritten, or have access to the object while its components are still in the free pool. If the latter occurred the components could be allocated to another object later on and the access residual caused by the crash would allow the new object to be accessed by the other user.

Conclusion

This paper has described what object reuse is and explained the TCSEC requirements in this area. It discussed how objects are created and destroyed and what mechanisms need to be examined to verify that proper object reuse processing is performed. A method for conducting an object reuse study that verifies whether a system satisfies the TCSEC requirements for Object Reuse was presented and explained in some detail. Other areas that should be investigated but fell outside of the methodology were presented as well.

It is hoped that the material presented here will help educate the vendor and evaluation community about object reuse and what the major issues and concerns are in this area. The methodology is intended to be a guide to vendors trying to verify the effectiveness of their product's object reuse mechanisms and for evaluators as well, as a metric with which to judge the thoroughness and completeness of a vendor's verification efforts in the object reuse area. The author hopes that the understanding of object reuse provided here will help clarify an area of concern for trusted systems that has received little published attention and assist the security community in developing and maintaining products with a higher level of assurance.

References

[1] M. Bach, *The Design of the Unix Operating System,* Prentice Hall Software Series, 1986.

[2] *Department of Defense Trusted Computer Security Evaluation Criteria DoD 5200.28-STD,* National Computer Security Center, December 1985.

[3] Dennis Hollingsworth and Richard Bisbey II, "Protection Errors in Operating Systems", UCS Information Sciences Institute, Marina del Rey, CA, June 1976.

THE DETERRENT VALUE OF NATURAL CHANGE IN A STATISTICAL DATABASE

Elizabeth A. Unger[1]
Professor
Computing and Information Sciences
Nichols Hall
Kansas State University
Manhattan, Kansas 66506

Sallie Keller-McNulty
Associate Professor
Statistics
Dickens Hall
Kansas State University
Manhattan, Kansas 66506

Abstract - Databases can be compromised inferentially using a variety of tools or methods. Previous literature has shown the difficulty of protecting a database from attacks using these tools and methods. The study of most inferential attack mechanisms has assumed the environment of concern is a static statistical database, a database where the anticipated results are statistical measures for groups of records and in which the unique record identifiers are not available to the user. Static means that the database remains constant over an interval of time in which the inferential attack takes place. In practice databases typically change over time, hence are dynamic. In this paper the results of an initial study of the deterrent value of natural change in a statistical database are given. The goal of this work is to define a measure of security for a statistical database as a function of the natural rate of change of the data, the dimensionality of the data, and the size of the database

1. Introduction

Providing access to data always carries the risk of disclosure of confidential or sensitive information. Security mechanisms applied to statistical databases attempt to minimize this risk while still providing useful information. The basic inference problem in statistical databases refers to the ability of an attacker, frequently a legitimate user, to infer sensitive information based on their view of the database and on information external to the database. Study of inferential methods of compromise and the deterrent value of several control mechanisms against these attacks has focused on a static database environment. A number of inferential methods have been identified. These include linear system attacks [1], [2], [3], [4], a subclass of linear system attacks known as trackers [5], [6], median attacks [6], [7], and insertion and deletion attacks [8], [9]. Some of the control mechanisms studied in a static environment include query set size restriction [10], cell suppression [11], [12], [13], partitioning [14], [15], [16], data masking or perturbation [17], [18], [19], [20], [21], [22], [23], random sample query sets [6], [24, [25] and data swapping [26], [27], [28]. The measures of security of the data are usually simple probabilities of compromise. Duncan and Lambert [29], [30] have proposed that measures of disclosure risk be computed by modelling the decision process of an attacker or "spy".

This paper investigates the effects of a dynamic database on standard inferential method called trackers. The use of a tracker is analogous to the manipulation of the cells in a cross-tabulation table to disclose information about a particular subgroup. The probability of compromise is the situation that the database may change during the execution of the queries required for the inferential attack is quantified. If no security mechanisms other than query set size restriction are applied to a static database, the probability of compromise is one, (assuming the target record set can be characterized and a tracker can be found -- Schlorer [27] has shown these are generally easily found.) In a dynamic database, the natural change reduces the probability of compromise without additional control mechanisms. The probability of compromise becomes a function of the rate of change, the dimensionality of the

[1] Visiting Professor, Computer Science- Telecommunications Program, The University of Missouri at Kansas City when work was completed

database, and the size of the database. Using this probability to form a measure of the security of the database, a database administrator or data-broker could determine what additional control mechanisms need to be applied to the database. The effects of a database change on the compromise control mechanisms also needs to be examined.

This paper is limited to a study of the use of a general tracker as the form of attack and query set size restriction as the only control mechanism. The dynamics of the database are resurrected to changes in attribute values of existing records. Within this environment, measures of security are developed.

2. Database model

The database model defined by Denning et al. [5] is used as the basis for this study. A database contains N records of m attributes. The attributes in a record can be divided into those in the user's view and those not in the user's view. A typical attribute not in a user's view would be one which uniquely identifies a record. In a statistical mode, the database can be queried using statistical function q(C), where C is a logical formula specifying the records to be summarized. Some commonly used statistical functions are sum, count, mean, median, standard deviation, etc. The Statistical functions considered here are limited queries involving sum and count functions.

It is assumed that query set size restrictions apply and only those queries whose size, number of records, are within the restricted range [k, N-k] will be retrieved, where k is the smallest set size retrievable. Specifying C involves associating values with the $|C|$ attributes in C. The set of records under attack are called target records and are defined to be query sets smaller that k or larger than N-k. The logical formulas used to define the query sets as well as a set of target records involve the logical operators AND, OR, and NOT (‾).

A general tracker is a logical formula T whose query set size is in the range [2k, N-2k], where $1 < k \leq N/4$. the values of un-retrievable queries with query set sizes smaller than k defined by the logical formula C can be obtained through the following manipulation of answerable queries:

$$q(C) = q1 + q2 - Q,$$

where

$$q1 = q\ (C \text{ OR } T),$$
$$q2 = q\ (C \text{ OR } \overline{T}),$$
$$Q = q\ (T) + q\ (\overline{T}).$$

Similar formulas exist for query set sizes which exceed the upper limit of the answerable query set range., Once a general tracker T and a logical formula C have been identified for some set of target records, the probability of compromise in a static database is one.

If the assumption that the database is static is removed allowing updates tò the values of attributes in existing records to change during the interval of time in which the queries necessary to fulfill the general tracker are made, the probability of compromise may be less that one. The fact that a compromise has not occurred may not be obvious to the attacker. With dynamic changes, the series of queries may lead to a plausible value for q(C), such a a one in a count query, but may not be the target set initially sought. This situation will be termed *apparent compromise*.If the true value for q(C) at the time the attack is completed returned obtained from the series of queries, then a *true compromise* said to have occurred. The probabilities of compromise present in this paper refer to true compromise.

The specific assumptions made with respect to the dynamic nature of the database are the following: a record update consists of changing the value of one attribute in a record; all changes occur completely at random; the tracker T has only one attribute and that attribute has but two values; the queries q(T) and q($\overline{T}$) are executed sequentially under the same state of the database hence Q is treated as a single query.

3. Probability of Compromise

The effect of single attribute value changes on the probability of obtaining a compromise with a general tracker is derived under the assumptions given in section II. It is also assumed that the maximum number of changes that occur between queries is one. The effects of these assumptions are that the probabilities derived will represent approximate deterrent values of natural change.

Let *p* denote the probability that a change occurs between two queries. During the sequence of three queries needed for the general tracker, 0, 1, or 2 changes can occur with probabilities (1-p), 2p(1-p), and p, respectively. It is assumed that a change immediately prior to the first query or a change immediately after the last query does not affect the ability to compromise the data. Only changes that occur during the time the three queries are processed affect the outcome. The analysis of the effect of updates on the general tracker must take into account all possible sequences of the three queries and all possible times at which the updates can be made during the execution of a query sequence. There are six possible query sequences for the three queries Q, q1, and q2.,. Updates to the database can occur between any two queries.

The deterrent value of natural change on the general tracker is considered in two ways. In section 3.1 it will be assumed that the updates only occur in attribute fields for which a compromise is sought, that is the sum or count is to be computed. In this case, which record is changed is analyzed. The second method, discussed in section 3.2, looks at the effect of updates moving records in and out of the set of records defined by the logical formula. For records to move among sets, categorical valued attributes used in the logical formulas must be changed.

3.1 Record Updates

The following Venn diagram of the general tracker query space will be used in this analysis.

	C	$\overline{C}$
T	u	w
$\overline{T}$	v	x

$$q(C) = q1 + q2 - Q$$
$$q(1) = (C \text{ OR } T)$$
$$q(2) = q(C \text{ OR } \overline{T})$$
$$(u+v) = (u+v+w) + (u+v+x) - [(u+w) + (v+x)]$$
$$2k < u+w < N - 2k; \; 2k < v+x < N-2k$$
$$u+v+w+x = N = \text{total number of records in database;}$$
$$u+v = t = \text{size of target record set.}$$

Figure 1. Venn Diagram of the General Tracker Query Space

If zero updates occur during the sequence of queries, the probability of compromise under this condition is 1. Therefore, P(compromise and no changes) = $1 \times (1-p)^2$. (1)

Exactly one update during the sequence can occur in twelve ways. Let ! between two queries denote that a change to the database has occurred between these two queries. Fro exactly on update Table 1 gives the twelve sequences that can occur.

Table 1. Twelve query sequences (conditions) with Exactly One Record Update

(1) q1 ! q2 Q
(2) q1 ! Q q2
(3) q2 ! q1 Q
(4) q2 ! Q q1
(9) Q ! q1 q2
(6) Q ! q2 q1

(7) q1 q2 ! Q
(8) q2 q1 ! Q
(9) q1 Q ! q2
(10) Q q1 ! q2
(11) q2 Q ! q1
(12) Q q2 ! q1

The probability of compromise under each of these twelve conditions can be determined by referring to Table 2. or example, under Condition (1), only the records in (C AND T) can change and still allow compromise. Thus, the probability of compromise under Condition 1 is x/N, assuming every record in the database has an equally likely chance of being updated. Under Conditions 5 and 6, an update anywhere in the database will completely deter compromise. Table 2 gives the probabilities of compromise under each of the twelve conditions above.

Table 2. Probabilities of Compromise with Exactly one Record Update

Conditions	Probability of Compromise
q1 ! q2 Q or q1 ! Q q2	x/N
q2 ! q1 Q or q2 ! Q q1	w/N
Q ! q1 q2 or Q ! q2 q1	0
q1 q2 ! Q or q2 q1 ! Q	0
q1 Q ! q2 or Q q1 q2	(u+v+w)/N
Q q2 ! q1 or q2 Q ! q1	(u+v+x)/N

Assuming each of the twelve sequences in Table 1 are equally likely to occur,

$$\text{P(compromise and exactly one change)} = \frac{1}{3N} p(1-p)\,[x+w+(u+v+w)+(u+v+x)]$$

$$= \frac{2}{3} p(1-p)\,. \qquad (2)$$

Exactly two updates during the sequence can occur in six ways. These six conditions and their corresponding probabilities of compromise are given in Table 3.

database, and the size of the database. Using this probability to form a measure of the security of the

Assuming each of the sex sequences in Table 3 are equally likely to occur,

$$\text{P(compromise and exactly two changes)} = \frac{1}{6N^2} p^2\,[x(u+v+w)+w(u+v+x)]$$

$$=\frac{1}{6N^2}p^2[t(N-t)+2wx] \qquad (3)$$

From equations 1, 2 and 3, the overall probability of compromise under this problem definition is:

$$\text{P(compromise)} = 1(1\text{-}p)^2+\frac{2}{3}p(1-p)+\frac{1}{6N^2}p^2[t(N-t)+2wx] \qquad (4)$$

Table 3. Probabilities of Compromise with exactly Two Record Updates

Conditions	Probability of Compromise
q1 ! q2 ! Q or q2 ! q1 ! Q	0
q1 !Q !q2	$x(u+v+w)/N^2$
q2 ! Q !q1	$w(u+v+x)/N^2$
Q ! q1 !q2 or Q ! q2 !q1	0

Since w+x=N-t, equation 4 is maximized with $w=\left\lceil \frac{(N-t)}{2} \right\rceil$ and $x=\left\lfloor \frac{(N-2)}{2} \right\rfloor$ (or vice versa). For ease of presentation, let w=x=(N-2)/2. Then,

$$\text{P(compromise)} < 1+\frac{5}{12}p^2-\frac{4}{3}p-\frac{\left(t^2p^2\right)}{12N^2} < 1+\frac{5}{12}p^2-\frac{4}{3}p. \qquad (5)$$

database, and the size of the database. Using this probability to form a measure of the security of the

The deterrent value of natural change is computed as 1-P(compromise) and is bounded below by minus the right hand side of Equation 5. Table 4 gives bounds on the deterrent value of natural change for different values of p. The value of p represents the rate of change of attribute values in the database. If p=1/2, then no change is as equally likely to occur between any two queries as one change.

Table 4. Minimum Deterrent Value of Natural Change Against the General Tracker with Record Updates

p	Minimum Deterrent Value ($-5/12\ p^2 + 4/3\ p$)
.10	0.1292
.20	0.2500
.30	0.3625
.40	0.4667
.50	0.5625
.60	0.6500
.70	0.7292
.80	0.8000
.90	0.8625

3.2 Updates on Categorical Valued Attributes

In this situation, to derive the probability of compromise, only the effects of updates on C an T will be considered, All attributes in the databases are equally likely to change. Table 5 is a comprehensive list of the types of updates that can occur and their corresponding probability of occurrence. Recall that T is restricted to only one attribute. It is assumed that changing the value of an attribute in C in a target record transforms that record out of the target record set and changing the value of an attribute in C in a nontarget record does not transform that record into a target record.

Table 5. Types of Updates Specified by Location of Attribute Changed

	Updated Attribute	Updated Attribute	Tracker Attribute	Updated Attribute	Probability
Update Type	in C	in T	in C $T \subset C$	in target record set	of Occurrence
1	yes	yes	yes	yes	$\lambda t/(MN)$
2	yes	yes	yes	no	$\lambda(N-t)/(MN)$
3	yes	no	yes	yes	$\lambda t(\lvert C\rvert-1)(N-t)/(MN)$
4	yes	no	yes	no	$\lambda(\lvert C\rvert-1)(N-t)/(MN)$
5	yes	no	no	yes	$(1-\lambda)t\lvert C\rvert/(MN)$
6	yes	no	no	no	$(1-\lambda)\lvert C\rvert(N-t)/(MN)$
7	no	yes	no	yes	$(1-\lambda)t/(MN)$
8	no	yes	no	no	$(1-\lambda)(N-t)/(MN)$
9	no	no	yes	yes	$\lambda t(M-\lvert C\rvert)/(MN)$
10	no	no	yes	no	$\lambda(M-\lvert C\rvert)(N-t)/(MN)$
11	no	no	no	yes	$(1-\lambda)t(M-\lvert C\rvert-1)/(MN)$
12	no	no	no	no	$(1-\lambda)(M-\lvert C\rvert-1)(N-t)/(MN)$

Note: Probability of occurrence is based on completely random change. N= number of records in the database and M= number of attributes in the database, $|C|$ = number of attributes in C, t= number of records in target record set, and + $P(T \subset C)$.

As indicated in Table 5, the probability of occurrence is a function of the size of the database, the number of attributes, the number of attributes in C, the complexity of the logical formula C, and the size of the target record set. In general, for a database with moderate N and M $|C|$ small relative to M and t small relative of N, Update Types 10 and 12 are most likely and Update Types 1, 3, and 7 are least likely to occur.

To analyze the probability of compromise in this second problem definition., the probability of compromise under each update type in Table 5 needs to be derived. For, example, consider Update Type 1 under the condition of exactly one change during the sequence of three queries. Assume any of the twelve query sequences given in Table 5. If the change is of Update Type 1, then the query set q1=q(C OR T) loses the changed record. The query sets of Q and q2 a re not affected by the change because the hanged record will be lost in both T and C but then gained back in the $\overline{T}$ argument in both of these queries. For a true compromise, the final result of q(C) must reflect the state of the database at the end of the sequence of queries. Consequently, the update must occur before the execution of q1 form compromise. There are six out of the twelve sequences in Table 1 that have an update before q1. Therefore,

$$P(\text{compromise and exactly one update of Type 1}) = \frac{6}{12}\left[2p(1-p)\right]\frac{(\lambda t)}{NM} = p(1-p)\frac{(\lambda t)}{NM}.$$

Similar arguments can be made for all twelve update types in Table 5. These results are given in Table 6.

Table 6. Probability of Compromise by Update Type

Update Type	Probability of Compromise and Exactly One Update	Probability of Compromise and Exactly two Updates of the Same Kind
1	$p(1-p)\ \lambda t/(NM)$	$p^2\ \lambda t/(3MN)$
2	$2p(1-p)\lambda\ (N-t)/(3MN)$	0
3	$p(1-p)\lambda\ t(\lvert C\rvert -1)/(MN)$	$p^2\lambda\ t(\lvert C\rvert-1)/(3MN)$
4	$2p(1-p)\lambda\ t(\lvert C\rvert-1)(N-t)/(MN)$	$p^2\ \lambda t(\lvert C\rvert-1)(N-t)/(MN)$
5	$p(1-p)\lambda\ t\lvert C\rvert/(MN)$	$p^2\lambda\ t\lvert C\rvert/(6MN)$
6	$2p(1-p)\lambda\ t\lvert C\rvert(N-t)/(MN)$	$p^2\ \lambda t(\lvert C\rvert-1)(N-t)/(MN)$
7	$2p(1-p)\lambda\ t/(MN)$	$p^2\ \lambda t/(MN)$
8	$2p(1-p)\lambda\ (N-t)/(3MN)$	0
9	$2p(1-p)\lambda\ (M-\lvert C\rvert)/(3MN)$	0
10	$2p(1-p)\ \lambda t(M-\lvert C\rvert)(N-t)/(3MN)$	$p^2\ \lambda t(M-\lvert C\rvert)(N-t)/(6MN)$
11	$2p(1-p)\ \lambda t(M-\lvert C\rvert-1)/(3MN)$	0
12	$2p(1-p)\ \lambda t(M-\lvert C\rvert-1)(N-t)/(3MN)$	$p^2\ \lambda t(M-\lvert C\rvert)(N-t)/(6MN)$

Note: For exactly two updates it is assumed both updates are of the same type. N= number of records in the database, M= number of attributes in the dates, $|C|$ = number of attributes in C, t = number of records in target record set, and $\lambda = P(T \subset C$.

The overall probability of compromise is $(1\text{-}p)^2$, for the probability of no change, plus the sum of the entries in both columns of Table 6. The corresponding deterrent value would be one minus this probability. To simply, assume N and M are large relative to $|C|$ and t. Under this assumption, the probabilities for Update Types 1 through 9 and 11 are approximately zero. The cumulative probabilities fro Update Types 10 and 12 are approximately p^2 and 2p(1-p)t/3 for exactly two and exactly one changes, respectively. The deterrent value of natural change computed under these conditions is:

$$\text{DeterrentValue} \approx 1 - \left[(1-p)^2 + 4p(1-p)\frac{t}{3} + p^2\frac{t}{6}\right].$$

Table 7 gives some approximate deterrent values of natural change for different values of p with t=1 and λ=1/2.

Table 7. Deterrent Value of Natural Change Against the General Tracker with Updates on Categorical Valued Attributes; N and M Large, t=1 and $\lambda=1/2$.

p	Approximate Deterrent Value $1-[(1-p)^2+2p(1-p)/3+p^2/6$
.10	0.1283
.20	0.2467
.30	0.3550
.40	0.4533
.50	0.5417
.60	0.6200
.70	0.6883
.80	0.7467
.90	0.7950

4. Conclusions

The analyses indicate that natural dynamics in a database aids the minimum query set size control mechanism in the deterrence of compromise by the general tracker. Some common conclusions can be drawn from both the analyzes. The rate of change in the database, characterized by p, and the number of records, N, in the database both influence the probability of compromise. This can be seen by examining Equation 5 for the record update representation of dynamic change and by examining Table 6 for the update on categorical valued attribute representation. In both cases, as N increases and as p increases, the probability of compromise decreases.

The analysis based on the dynamic database representation of updates on categorical valued attributes lead to some additional conclusions. Specifically the relationship characterized by λ between attributes defining the tracker, T, and the logical formula C and the dimensionality of C, both affect the probability of compromise. This means that if attributes that are likely candidates for trackers can be identified a control mechanism could be designed that uses this information. The number of records in the target set also influences the probability of compromise. As one might expect, it is more likely for compromise to occur over larger aggregates of the data, i.e., t large.

This study is limited in that several assumptions were necessary to carry out the derivations. It points out that modeling dynamic change in a database in not trivial. The effects of dynamic change on other forms of compromise needs to be investigate. The deterrent value of natural change in the face of other control mechanisms needs to be measured. It is these measures of deterrent value that a database administrator should use as a baseline measure of security for the database.

5. References

[1] Beck, L.L., "A Security Mechanism for Statistical Databases," *ACM Transactions on Database Systems*, 5(3), 1980, pp. 316-338.

[2] Causey, B. D., Cox, L. H., and Ernst, L. R., "Applications of Transportation Theory to Statistical

Problems," *Journal of the American Statistical Association*, 80, 1985, pp. 903-909.

[3] Chin, F. Y. and Ozsoyglu, G., "Auditing and Inference Control in Statistical Databases," University of California, San Diego, 1980.

[4] Cox, L. H., Suppression Methodology and Statistical Disclosure Control," *Journal of American Statistical Association*, 75, 1980, pp. 377-385.

[5] Cox, L. H., "Linear Sensitivity Measures in Statistical Disclosure Control," *Journal of Statistical Planning and Inference, 5*. 1981, pp. 153-164.

[6] Cox, L. H., "Controlled Rounding," INFOR, 20(40, 1982, pp. 423-432.

[7] Cox, L.H. "A Constructive Procedure for Unbiased Controlled Rounding" *Journal of the American Statistical Association*, 82, 1987, pp 520-524.

[8] Cox, L. H., Fagan, J. Greenberg, B. and Hemmig, R., "Research at the Census Bureau into Disclosure Avoidance Techniques for Tabular Data," *American Statistical Association 1986 Proceedings of the Section on Survey Research Methods*, 1986, pp. 388-393.

[9] Dalenius, T., "Towards a Methodology for Statistical Disclosure Control," *Statistisk Tidskrift,* 5, 1977, pp 429-444.

[10] Dalenius T. and Denning, D. E., "A Hybrid Scheme for Release of Statistics'" *Statistisk Tidskrift,* 2, 1982, pp. 97-102.

[11] Dalenius, T. and Reiss, S. P., "Data-Swapping: A Technique for Disclosure Control," *Journal of Statistical Planning and Inference*, 6, 1982, pp.73-85.

[12] Davida, G. i., Lipton, D. J., Szelag, C. R., and Wells, D.L., "Data Base Security," *IEEE Transactions on Software Engineering*, SE-4(6), 1978, pp. 531-533.

[13] DeMillo, R. A. Dobkin, D. P., and Lipton, R. J., "Even Databases That Lie Can Be Compromised," *IEEE Transactions on Software Engineering*, SE-4, 1977, pp.73-73.

[14] DeMillo, R. A., and Dobkin, D. P., "Combinatorial Inference," Foundations of Secure Computation, Academic Press, New York, 1978.

[15] Denning, D. E., "Secure Statistical Databases with Random Sample Queries," *ACM Transactions on Databases Systems,* 5(3), 1980, 291-315.
16] Denning, D. E., and Denning, P. J., "The Tracker: A Threat to Statistical Database Security," *ACM Transactions on Database Systems*, 4(1), 1979, pp. 76-96.

[17] Dobkin, D. Jones, A. K., and Lipton, R. J., "Secure Databases: Protection Against User Inference," *ACM Transactions on Database Systems,* 4(1), 1979, pp. 97-106.

[18] Duncan, G. and Lambert, D., "Disclosure-Limited Data Dissemination," *Journal of the Ameri-*

can Statistical Association, 81, 1986, pp.10-18.

[19] Duncan G. and Lambert D., "The Risk of Disclosure for Microdata," *Journal of Business and Statistics*, to appear.

[20] Fiege, E. L. and Watts, H. W., "Protection of Privacy Through Microaggregation, "in *Data Bases, Computers, and Social Sciences,* Wiley-Interscience, New York.

[21] Hoffman, L. J. and Miller, W. F., modern methods for Computing Security and Privacy, Prentice-Hall, Englewood Cliffs, N.J, 1977.

[22] March, M.J. and Norris, D. A., "Disclosure Avoidance Techniques in the Canadian Censuses of Population and Agriculture, "*American Statistical Association 1987 Proceedings on the section on Survey Research Methods*, 1987, pp233-238.

[23] McLeish, M., "Further Results on he Security of Partitioned Dynamic Statistical Databases," *ACM Transactions on Database Systems*, 14(1), 1989, pp98-113.

[24] Liu, L., "On Linear Queries in Statistical Databases," The MITRE Corp., Bedford, Mass. 1980.

[25] Schlorer, J. "Disclosure from Statistical Databases: Quantitative Aspects of Trackers," *ACM Transactions on Database Systems,* 5(4), 1980, pp467-492.

[26] Spruill, N. L., "The Confidentiality and Analytic Usefulness of masked Business Micro data, "*American Statistical Association 1983 Proceedings of the Section on Survey Research methods*, 1983, pp602-613.

[27] Strudler, M., Lock Oh, H. and Scheuren, F., "Protection of Taxpayer Confidentiality With Respect to the Tax model, "*American Statistical Association 1986 Proceedings of the Section on Survey Research Methods*, 1986, pp375-381.

[28] Tendick, P. and Matloff, N. S., "Recent Results on the Noise Addition Method for Database Security, "*American Statistical Association 1987 Proceedings for the Section on Survey Research Methods,* 1987, pp406-409.

[29] Wien, S, "A Security Mechanism Based on Uniform Rounding to Prevent Tracker Techniques, 'COMPSTAT 1986, Physica-Verlag, Vienna, 1986, pp460-465.

[30] Yu, C. T. and Chin, F. Y., "A Study on the Protection of Statistical Databases, "*ACM SIGMOD international Conference on the Management of Data*, 1977, pp169-181.

EXPERIENCES IN ACQUIRING AND DEVELOPING SECURE COMMUNICATIONS-COMPUTER SYSTEMS

Captain Charles R. Pierce

Air Force Cryptologic Support Center (AFCSC)
San Antonio, Texas 78243-5000

ABSTRACT

This paper describe experiences in acquiring secure computer systems. The integration of the DOD Trusted Computer System Evaluation Criteria and other policy documents into the acquisition process is ill defined and poorly performed. This paper hopes to provide some observations and lessons learned that can provide program managers with help in improving the acquisition and development process. It provides neither a description of the ideal process for procuring secure systems nor does it take a complete view of the system life cycle.

Introduction

The primary focus of this paper is the integration of computer security (COMPUSEC) into systems acquisition. It will concentrate on COMPUSEC and avoid details on the inclusion of Communications Security (COMSEC) and Emanations Security (TEMPEST) in system development. However, it is neither possible nor feasible to efficiently pursue the development of secure systems without ensuring the adequate inclusion of all related security disciplines in the effort. The need for security expertise in all three disciplines is vital for ensuring all inclusive security in systems acquisition or development agencies. Usually persons with extensive security expertise are not widely versed in the particulars of system acquisition or development. Conversely, acquisition personnel are most often not security specialists, nor are such specialists available in their organizations. This paper, produced by security specialists, attempts to bring some expertise in that area to acquisition personnel. To fully define or describe how to procure secure systems requires significant explanation and discussion and is beyond the scope of this paper. This is merely an introduction to what will be a series of papers discussing secure systems development. Topics for other papers include defining a secure environment and developing a related security policy, translating a security policy into specifications and requirements, performing source selection for secure systems, developing applications software in the secure environment, performing security test and evaluation (ST&E) in the operational test and evaluation (OT&E) process, system certification, accreditation from the Designated Approving Authority's (DAA) point of view, and many others. This is but an overview of each of these areas and an introduction to many resulting problems.

Acquisition personnel are usually concerned with a system's development up to that point in the life cycle when management responsibility for the system is passed to its user or maintainer. To provide a properly secure system the acquirers must assure the end users that all of their concerns have been comprehensively addressed. This is our goal, to ensure complete life cycle coverage of security requirements. The primary concentration will be in the early phases of the life cycle, the primary concern of acquirers or developers. No less important are the operation and maintenance phases, but these will remain detailed topics for another paper.

A well structured program for ensuring life cycle security embraces the use of properly performed risk management. Risk management, composed of risk analysis, certification, and accreditation, begins at the initialization of the life cycle when the system's sensitivity (e.g., information classification levels) and criticality (i.e., mission reliance) are defined. The risk analysis phases of risk assessment and economic assessment will be constantly repeated as iterations of test and evaluation provide reliable measures of the adequacy of implemented security measures. The developers final security action is to provide certification to the end user that security requirements have been met by the implemented security measures. This certification, including the appropriate documentation, allows the user to accredit (approve) the system for operation. The process doesn't stop here however, because the user must maintain the accreditation throughout the life cycle. Thus, ensure that guidance provided for the early life cycle phases is easily transferable to the end user and the later stages of the complete life cycle.

Planning for Procurement

The availability of COMPUSEC support for a system's development depends greatly on the type of support requested and the difficulty in finding the necessary resources to provide that support. The source selection process in the DOD involves a considerable amount of time and effort, probably at locations away from the supporting organization. The source selection evaluation support for the Air Force's Standard Multiuser Small Computer Requirements Contract (SMSCRC) required 10 weeks of COMPUSEC support at Hanscom AFB in Boston, MA. On the contrary, requirements review and specifications development can be done on a desktop and via the mails or telecommunications, thus incurring little or no travel expense. The COMPUSEC supporter usually must bear the expense of providing support. However some types of support, e.g., security support in performing OT&E, can be separately funded by OT&E organizations, or a system's program management office (PMO) may have resources to provide for support. It is most important to define support requirements early in the system life cycle and determine the source for funding them. Providing support from requirements definition through test and evaluation to final certification requires many years of planning and performance as well as the maintaining of expertise continuity.

Although support requirements continue to grow, the ability to provide the support has not grown at a comparable rate. Currently the National Computer Security Center (NCSC) provides very limited support to specific acquisitions. For those in which it is involved the type of support provided is limited to specific areas, for example evaluating the Trusted Computing Base (TCB) or COMSEC measures. Seldom does NCSC participate throughout a system's life cycle from requirements definition to final accreditation. A system PMO is usually staffed to meet acquisition needs but seldom has the needed security expertise. Nor is there an easy way to train PMO personnel to a level to where they can stand without outside support. It's a wise program manager who plans for applying his own resources, COMPUSEC assistance, NCSC's TCB evaluation capabilities, and other organizational support at the proper phasing and timing during the development life cycle. So how's the program manager suppose to be so smart? Developing some of that wisdoms is the goal of this paper.

Support Up to Contract Award

In the Air Force a COMPUSEC analyst will use DODD 5200.28 [6], AFR 205-16 [2], CSC-STD-003-85 [5], and risk analysis results to translate user requirements into a brief security policy. This policy statement then becomes the basis for developing a flexible, high level security architecture. This architecture is then used to produce the system's technical and operational requirements and specifications. The risk analysis, policy, architecture, and requirements all are living documents which must be constantly revisited and updated as the life cycle progresses. None are set in concrete.

The TCB specifications for some systems have been developed from a Mitre Corporation technical report, MTR-9673, Trusted Computer System Security Requirements Guide for DOD Applications [8], developed for the NCSC. This report translates the criteria of DOD 5200.28-STD [7] into the appropriate language for a Request for Proposal (RFP). It provides sample statements for "A" type System Specifications, Statements of Work (SOW), and Contract Data Requirements List (CDRL) items and related Data Item Descriptions (DID). The technical report's baseline statements were tailored to meet specific user requirements and to include contractor tasking to provide certification and accreditation support. There is some controversy as to whether such extensive TCB specifications should be included in a RFP. Simple logic implies that merely asking for an appropriate TCB level should be sufficient. More on this later.

Evaluating competing TCB proposals in source selection is not a well defined process. It relies primarily on the availability and experience of security evaluators. Because the evaluator does not have anywhere near the time to fully evaluate each proposed TCB (a process that would result in each being issued a rating) he must make many value judgements based solely on limited experience. Much of the evaluator's time is spent in getting clarifications from the proposers on exactly what they are proposing. It also becomes fairly obvious that there are few vendors who know how to properly propose a TCB!

Developmental Support

After a contract is awarded, the support required changes from generating acquisition products to essentially a verification and validation effort. For contract deliverables, ensure that what is delivered is that which was requested and that the product's quality is as specified. Program management personnel not experienced in security may not be able to tell if the deliver Security Features User Guide meets user requirements. In fact, the user may not be able to tell! This is also when the user learns that it would have been appropriate to include user training in the TCB specification. It becomes evident that the COMPUSEC analyst must become very familiar with both the system being supported and the mission it supports.

It is increasingly important to be sure security is adequately addressed in each related deliverable and in each standard design review.

For one reason or another the program office or user is sometimes requested to back off of security requirements. Developers may suddenly find they can't meet schedules or they underestimated the effort involved. It's extremely critical not to back off the security requirement. Proper security is as essential as any other technical or operational requirement. Effective security is as essential to a computer system as is computational speed. The COMPUSEC supporting agency becomes the safety valve to ensure security is not weakened.

Because TCB development is in its relative infancy, it is not likely very many already evaluated systems will be available at contract award time. The PMO and user should be prepared to accept a phased development of the TCB within the standard life cycle. The phasing should be adjusted such that the final rating for the system is achieved before Program Management Responsibility Transfer (PMRT) or delivery to the system supporter or user.

Security Test and Evaluation (ST&E)

The various forms of test and evaluation provide the final opportunity to verify the successful implementation of security measures against security requirements. ST&E is not usually considered as part of the formal DOD T&E process, e.g., Developmental Test and Evaluation (DT&E). Nor is it a particular part of the TCB development process. In reality ST&E can be performed as a part of any of the various T&Es or as a standalone process. ST&E for the TCB can be performed as part of the system DT&E or in the commercial product evaluation process of the NCSC. Testing the integration of the TCB into the operating environment, with the required applications software, can be specified as a requirement and thus Operational Test and Evaluation (OT&E) can provide the output for the ST&E. For a non-developmental system (off-the-shelf), testing such as DOD's Qualification Test and Evaluation (QT&E) or Qualification OT&E (QOT&E) can meet ST&E needs, although it is unlikely any TCB could be evaluated. The point is there is a variety of contract deliverable or independently performed tests that can meet ST&E needs without performing it as a separate process. Be flexible. ST&E tasks can be widely dispersed with the PMO providing the final "ST&E Report" gleaned from the various tests. Or the ST&E report could be tasked to the contractor. Or an independent agency can perform a separate ST&E. We go back to the maintenance of those initial products, the system risk analysis, security policy, architecture, and requirements to find if all needed plans, responsibilities, and actions were properly implemented to assure adequate security is delivered.

Lessons Learned in Previous Acquisitions

As previously mentioned there is some disagreement as to the how to specify a TCB in an RFP. If the Evaluated Products List (EPL) [9] were fully populated with a wide range of systems in each class, it could be practical to specify simply a class level with the caveat that the TCB must also meet all other system requirements. The EPL is not currently populated to such an extent nor is this likely before the turn of the century, if systems continue to appear at the current rate. Thus it seems practical to continue to fully specify the TCB requirements, perhaps by continuing to use the MTR-9673 or its successor. Additionally any required extensions to a standard TCB class must also be included in the specification. This would include such items as an informal model or limited security administrator support at the C2 level. As was done with the SMSCRC, a statement can be included that states that an available TCB may be proposed but that it must also meet the RFP specification requirements, some of which were not part of the B1 criteria. A quick glance at the reports for products currently on the EPL reveals that in nearly all cases each product exceeds the minimal set of criteria for the class it was awarded.

Because neither the program office nor user normally have the time or expertise to perform most certification and accreditation activities, it is extremely practical to include these activities as contract deliverables. In addition to reducing the PMO and user's workload this shifts the bulk of these tasks to the contractor, who most likely will have the knowledge and resources to perform the tasks. Much of the documentation required for certification or accreditation must be developed partially or in total to meet the TCB evaluation requirements as it is. Proper CDRLs and DIDs can ensure the deliverables meet all requirements. The actual responsibility for certification must remain with the PMO, and the user still accredits the system. These duties cannot be delegated to the contractor.

The appropriate CDRLs and DIDs for computer security traditionally have not been available. Those that have been used, even in conjunction with the MTR-9673, required extensive tailoring. Nor did they fit well into the standard DOD life cycles. The user was required to request something not easily defined and the PMO was left with little guidance to help the user. The practical solution became "hope for the best." However, if nothing is called for, that's exactly what you'll get. This is an area requiring a great deal of flexibility. If, for example, a DID calls a Trusted Facility Manual in a particular format, it is probably more beneficial to accept a previously developed commercial product in a different format. This avoids the development cost and time for that product.

The timing of item deliveries can have a marked impact on the cost of security. The documentation for a TCB is part of that TCB and, even though it may have been listed under separate CDRLs, it must be delivered as part of the TCB and not as separately priced items. However, if the TCB development requires an extended time period such that its certification will be delayed until after the system becomes operational you may wish to call the CDRL items separately to support the system's certification and accreditation. This is a decision that should be planned for and discussed with the DAA as it will probably be derived from a DAA requirement.

For past developments, many similar current acquisitions, and RFPs not yet released TCBs are specified as delayed deliverables. For the near term, multilevel TCBs will probably continue to be delayed deliverables due to the lack of available systems at the B and A levels. Until the EPL becomes fully populated it is highly unlikely that B or A level systems will be available at contract award time. Both procuring agencies and users must determine the appropriate phasing of TCB delivery and how much delay to tolerate. Even though the delayed TCB would not be rated, it is still possible to certify, accredit, and operate the system in a secure manner. If the system delivered at contract award varies greatly from the eventual TCB, you can operate in a more restrictive mode of operations, for example, system high as opposed to multilevel. Be sure the vendor-provided certification plan includes the migration from that configuration to the eventual certified system. If the system delivered at contract award is essentially that which will be rated (it may even be under NCSC evaluation at the time), you can operate it as intended. Meeting the security requirements is the goal, not that the certificate of rating has been issued.

As has previously happened, you may wish to extend upper level TCB features or assurances to lower level systems. For example, the security policy for a C2 TCB for a sensitive but unclassified system would not provide the level of detail for managing sensitivity "levels," a B level feature intended for separating classification levels. However, you can specify that the security policy be developed such that it serves as and describes itself as a "classification guide" for the management and separation of unclassified "levels" based on discretionary access controls. The "policy" would describe how group or user controls will be set up and used to maintain the separation of "levels." As can be typical in many embedded applications with limited user interface, the required security features need only meet the C2 criteria for system high operations. An aircraft, for example, can have but a single user, the pilot, when in operation. But, because of the criticality of the supported mission, more assurances must be provided that the system is secure; for example, B2 testing and configuration management must be required. These "special cases" plus valid risk management cost benefit analyses provide further reasons for producing detailed specifications.

Much like extending TCB features to differing levels, new features can be added to TCB levels, except the new features are not part of the TCB or any current interpretation of the criteria. These added features may or may not be trusted. For example, a terminal control module was added to one B1 TCB to provide remote terminal controls. A fully implemented B2 TCB with a trusted path and device labeling and controlling, along with its accompanying longer development schedule and detailed requirements, was not needed but some remote terminal controls were. Terminals may not be physically located outside the system's controlled area but operations do require some terminal controls. The additional controlling module will is tested, certified, and accredited but not rated as part of the TCB. This implementation meets the system's policy, fits into the architecture, meets the specifications, and counteracts the vulnerabilities revealed in the risk analysis. Adding audit capabilities for a C2 system may be another practical application. These additional features, assurances, or documentation may be required by policy requirements other than DOD 5200.28-STD, for example the governing agency regulations, the system's policy, or as a result of risk analysis recommendations.

Observed Shortcomings in Acquisitions

Applications software is seldom, if ever, considered in the security specifications development. When security concerns arise, it is assumed that the TCB will manage all security. The affects of applications, and for that matter all interfaces, on the TCB are overlooked or ignored. This can become particularly worrisome if a data base management system (DBMS), network interface, or another discretionary or mandatory access policy violator is involved. Applications or interfaces must be designed to take advantage of the TCB security features, not implemented so as to violate these features. If not, security system redesign or retrofitting attempts become increasingly possible. Redesign or retrofitting is the worst possible event to encounter during development.

All eventual connections are not usually defined in the original specifications. This is because the requirements to connect these systems are not stated early in the life cycle. These connection requirements usually appear at the later stages of the life cycle. The system development has progress to where it is now too late to modify the system design or implementation. If the connectivity absolutely must be made, the TCB level and other implemented features no longer provide an adequate level of security. Late appearing connectivity requirements may result from the success of implementing the security of the basic system. Those wishing to connect to the basic system assume its secure system can also managed their security needs, most likely a false assumption.

A Concept of Operations, to include security, is seldom developed until the system nears initial operations. It then usually conflicts initially with the stated requirements. The basic failure stems from not involving the user early enough in the life cycle. The concept should specify the user's operational requirements such that the acquiring agency can convert them to technical specifications. Closer to the action, the user has a clearer view of what other security disciplines, e.g., physical or information will be in place or must be implemented regardless of the TCB level.

As with operations, maintenance concepts and requirements are completely overlooked as part of the security requirements. Maintenance needs drive multiple areas, including personnel clearances, maintenance facility clearance, storage, packing and shipping, disposal, clearing, and declassification (magnetic remanence). This also involves software maintenance for the TCB, including configuration management, and the applications. Each of these can affect maintaining the certification and accreditation.

Risk analysis is always in the catch up mode. Properly performed risk analysis should lead the effort, not be reactive to developing events. All security requirements should have flowed from risk analysis results. A reactive mode will force a last ditch effort to catch up that can overlook some risks or overpower available resources.

Valid cost information is lacking. The difference between TEMPEST and non-TEMPEST equipment can be cost compared. The cost difference between a B2 or A1 TCB cannot as yet. There is also a lack of history on TCB costs. Similar TCBs have been delivered at the same level, with one costing approximately $19,000 per copy while another came in at less than $300 per copy. This lack of valid numbers prevents the generation of valid cost-benefit ratios or returns on investment. It also causes problems when doing the economic assessment phase of the risk analysis.

Criticality requirements are difficult to interpret into specifications and therefore generally omitted. System integrity and assurance of service measures are not yet defined in any approved trusted criteria or standards. Each criticality requirement must be translated individually for each RFP, limited by the specifier's experience.

Perhaps the most glaring and far reaching problem is the failure to use a systematic security approach. Again, the TCB is assumed to satisfy all the security requirements. COMSEC and TEMPEST requirements may be overlooked or assumed away. In reality the best approach is usually a systematic approach. In nearly all cases, both military and commercial business, proven measures for personnel, physical, and other security needs are required anyway. Typically within DOD, uncleared people are seldom allowed access to classified systems. These features may also reduce the resulting level of TCB requirement. A particular area not usually covered is the building of "systems" from trusted components (which may be other trusted systems) or secure features not having a specific level of trust, for example, cryptographic equipment or trusted software.

Remedies and Requirements

Because AFCSC's mission is to provide appropriate, systematic security assistance for acquisitions, we have undertaken many efforts to meet the noted shortcomings and to improve our support from the lessons learned. Some of these efforts are nearly complete while others have recently begun. Although designed for the DOD environment, they are easily modifiable for non-DOD systems.

The Program Manager's Handbook for Security-Relevant Acquisitions [4] is an Air Force Special Security Memorandum (AFSSM) providing guidance for program managers on how to approach and include security in computer system acquisitions. Each technical security discipline is addressed as well as administrative and procedural methodologies. Major points are the process of certification and accreditation and the building of the security disciplines into them and the acquisition process. The integration of security or aspects of it into the acquisition life cycle is the major result.

Accompanying the Program Manager's Handbook are CDRLs and DIDs related to the various security disciplines. Where existing DIDs are sufficient for the state of the art they are included. For areas where deficiencies exist, new DIDs were developed and submitted for inclusion in the Acquisition Management Systems and Data Requirements Control Listing (AMSDL). The DIDs provide for documentation related to each of the discussed security disciplines and products, including the TCB deliverables, required to support certification, accreditation, and life cycle support.

Another AFSSM, Computer Security in the Acquisition Life Cycle [3], begins with the life cycle process in DOD-STD-2167A. It takes each life cycle phase and describes the actions a program manager must take and the security products affected in that phase. A series of charts provide for the flow of specific actions leading to accreditation. Key points on the charts indicate where DAA interaction or decisions are required. It also provides a summary chart of these key DAA points. It describes methodologies, tools, and techniques that can be used to develop secure systems and their applicability and timing in the life cycle.

Another effort is the is development of a series of products translating criticality requirements into acquisition guidance. Criticality is that part of computer security assuring the performance of critical mission support. This is usually implemented by assuring the integrity of the mission data or processes or assuring the system's service availability. The first product will be a set of criticality criteria designed as an interpretation of DOD 5200.28-STD. An accompanying interpretation describes how to apply both the criticality criteria and DOD 5200.28-STD to what are commonly called embedded systems. The primary focus is on those systems embedded in weapon systems although function, not size is the primary point of consideration. Thus, application to some ADP systems, for example a minicomputer used as a network controller, is possible. A follow on effort will involve translating the criticality criteria into an acquisition guideline, much like the Trusted Computer System Security Requirements Guide for DOD Applications.

To fill specific apparent spaces in the life cycle where security related actions are not performed or products not produced, we are developing a series of "gap fillers." The logical first product is a guide for forming a system's security policy derived from user requirements. From the policy flows all subsequent actions or products. (Although acquisition oriented, the process can apply to existing systems.) A special point of this effort is guidance for deriving a policy for developing secure systems from both trusted and untrusted components. The resulting policy would define a level of trust for the system in addition to other security requirements. If a specific level of trust is not practical, e.g., for an embedded system, then there is direction for building an accreditable policy.

After user requirements have been interpreted into a policy, which then becomes a specification and SOW in the RFP, competing vendors submit their proposals. The next major effort, with its own set of problems, is the technical evaluation of competing proposals in the source selection process. For personnel not experienced in TCB development, it may not be easy to tell if a proposal meets the technical specifications of the RFP. It is also difficult to determine qualitative differences between competing proposals. Another guideline will provide experience based guidance for performing these evaluations.

Following source selection and contract award, development begins in earnest. Following the Computer Security in the Acquisition Life Cycle guideline and good risk management practices leads to performing ST&E, the final step in the developmental risk analysis process. ST&E verifies the security implementation and leads to certification. An ST&E guideline will provide direction for including security in the existing T&E process and performing specific ST&E where existing T&E is not sufficient, for example, when application software is added to the system. Specific guidance provides for developing test objectives, measures of performance and results evaluation.

The Big Picture

An overall agency security policy must cover the integration of these and other products, with a hierarchical structure for using them, into the agency's security program. These products are far from providing complete and fully detailed guidance. They are the best derived from the described experiences. They are living documents that will require continual update much as do all life cycle documents. Work yet to be completed involves translating all security requirements into standard, or nearly standard, acquisition terms. Also, as secure systems begin to proliferate, present maintenance structures will be required to become adept at maintaining the secure systems. At this time few of the potentially affected organizations know this is about to happen. When we blindside them with the news, we must be ready to immediately provide them the capability to provide support or once again enter a time of retrenchment and catch up.

REFERENCES

[1] AFR 56-31, Security Policy and Requirements in the Development and Acquisition of Computer Systems (Draft).

[2] AFR 205-16, Computer Security Policy, 28 Apr 89.

[3] AFSSM 5010, Computer Security in the Acquisition Life Cycle (Draft)

[4] AFSSM 5024, Program Manager's Handbook for Security-Relevant Acquisitions (Draft)

[5] CSC-STD-003-85,Computer Security Requirements--Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, 25 Jun 85.

[6] DODD 5200.28, Security Requirements for Automated Information Systems (AIS), 21 Mar 88.

[7] DOD 5200.28-STD, DOD Trusted Computer System Evaluation Criteria, 26 Dec 85.

[8] MTR-9673, Trusted Computer System Security Requirements Guide for DOD Applications, Mitre Corporation, Bedford, MA.

[9] Information Systems Security Products and Services Catalog, Apr 90.

SECURE SYSTEMS INTEGRATOR: AN HONORABLE PROFESSION?

Virgil Gibson
Grumman Data Systems
6862 Elm St.
McLean, VA 22101

Abstract

The purpose of this paper is to describe the consulting, design and engineering services which are typically needed to integrate a system with significant security requirements. Secondarily, an emphasis will be placed on the importance of carefully selecting a security integrator, and the risks of attempting to integrate such a system without a qualified security architect.

The paper will include a short tutorial on the DoD Trusted Computer Systems Evaluation Criteria (TCSEC), the Product Evaluation Program, application of interpretations of the criteria and documents giving guidance on how to apply the criteria.

A survey of currently evaluated Computer Security products will be presented, along with a sampler of not evaluated products which provide significant security functionality, and some pitfalls which could result from their use without advice from a security architect. Some examples of ways, some sensible and some not so sensible, which have been used in the past to select Security software for large System-370 applications will also be explored.

A brief overview of the President's Council on Integrity and Efficiency major audit and results of their recommendations will enhance the argument of the paper. How did we get here? How can we avoid doing that again? The assertion of the paper is that only a broadly experienced expert can sift through the multitude of products, both evaluated and not evaluated, and mesh those with the constraints of environmental and functional requirements to arrive at an architecture which satisfies the customer and provides the level of security protection needed to allow smooth, "clear conscience" operations.

Introduction

In the majority of computer security publications and literature today, the main concern is with access control. The thing being controlled might be a terminal, a system, data within a system, or a certain application programs' files. The most common way of controlling access is through physical isolation, locks, barriers, etc.

When computers are part of the barrier, then a special technology gets invoked: technical computer security. The more sensitive the information to be protected by the computer, the more confidence must be placed in this technology. The DoD, through the National Computer Security Center (NCSC), has established a criteria to measure the strength of security products and a product evaluation program to provide information about the capabilities of various classes of security products.

THE "ORANGE" BOOK - Standards for Computer Security

The Trusted Computer System Evaluation Criteria (TCSEC) (aka DOD 5200.28-STD Dec.1985) defines the features, assurance and documentation which must be designed into a computing system if it is to be trusted to control access at all. The TCSEC describes divisions of progressively more robust protection:

Division D: Evaluated Subsystems which do not meet all the requirements for Division C.

Division C: Discretionary Access Protection. Consists of two classes, C1 and C2. Systems can be relied upon to enforce access determined by the owner of data and the IDentification of requestor.

Division B: Mandatory Access Protection. Consists of three classes B1-B3. Systems can be relied upon to enforce control according to Law or regulation (clearance).

Division A: Verified Protection. A single class A1. Provides mathematical proof that a system protects data at the highest level available within the state of the art.

The term Controlled Access Protection describes a system where access to data, programs, directories and such, is constrained to only those users with a "need to know" -- of course, the need to know is determined by the "owner" of the data (hence it's called discretionary access control). The familiarity of this term is understandable given the DoD mandate (DoD Dir 5200.28) that all affected systems will implement this capability by 1992. Controlled Access Protection in the TCSEC is class C2.

Since the concepts of Clearance and Classification are not dealt with at C2, it is clear that any classified processing in C2 systems will have to be done with all users cleared to the level of the highest information on the system.

THE CONTROLLED DISSEMINATION PARADOX

Getting information flow up the chain of command in this situation has been fairly easy to solve in separate systems, namely either develop a specialized guard or use the tried and true air-gap (sneaker net) or some other such method with a person or physical barrier to ensure that protected information couldn't leak down to systems of lower classification. Of course this poses a particularly difficult problem for the Commander whose function it is to get information - in the form of policies, orders, etc to flow down the chain of command.

The whole idea of a command center where information comes in from a variety of sources, is displayed and digested and acted upon, conflicts with traditional notions of security (i.e. access control). The action to be taken is usually in the form of orders, changes to procedures or whatever, which in order to be effective, must be disseminated broadly and rapidly. The security paradox which a command center embodies has been called controlled dissemination, a class of computer protection which is outside the criteria thus far defined.

THE ARCHITECTURE CHALLENGE

Designing an architecture for computer systems with security features is never trivial. Designing one with the major complexities found in a command center and with the security features necessary to provide true multilevel secure operations (where some users are not cleared for all information in the system) is a monumental effort. That is what controlled dissemination requires. And a qualified security architect is indispensable to the effort.

A couple years ago the list of evaluated products was so short (Figure 1) and so little research had been reported, that most computer security efforts were designed, built, tested, and certified by a single contractor. There are obvious advantages to this development method, especially when it comes time to blame somebody. The disadvantages are there too; cost and life cycle maintenance of special software come to mind.

THE CHINESE MENU

In today's more constrained fiscal climate "do-it-yourself with commercial off the shelf products" (DIY with COTS) is the catchphrase. The list of products being evaluated is extensive (Figure 2); and the list of unevaluated products vying for attention (Figure 3) is continually growing. The Trusted Network Interpretation (TNI) of the TCSEC has sparked much discussion of how a do-it-yourselfer can hook these various products together to form a system which will at least satisfy the accreditor's interconnection requirements. When something goes wrong getting help will be tough, and often too late.

It's better to start with a system integrator who's been there before, who can conceptualize the implementation from your requirements, and who can select an architecture which will meet power-processing needs and comply with security constraints using COTS evaluated products. This approach admittedly isn't as cost averting as DIY, but you know what it's going to cost up front; and the integrator will avoid the common pitfalls by having been there before.

COTS SECURITY COMPONENTS (1988)

DISCRETIONARY ACCESS PROTECTION
(Div C)
Access by Identification
RACF(MVS); ACF2; ACF2(VM); UTX/32S; Top Secret(MVS); Unisys A-Series; CDC/NOS

Mandatory Access Control (Div B)
Access by Regulation (Law Clearance)
Multics

Verified Protection (A1)
SCOMP

Figure 1

COTS Security Components:
Gordian; DPS-800-12; PFX PASSPORT; Watchdog PC; ACE; Sentinel; PC/DACS; Triad +, Safeword XENIX; Surekey; Onguard; IDX-50; Citadel; X-Lock 50; Tigersafe OR; Private Access L20; Codercard CPP-300; ACS-PCSM/ISM; MAC-310

Access by Identification:
RACF(MVS); RACF(VM); ACF2(MVS); ACF2(VM); AOS/VS; HIS UX/WS; CDC/NOS; VAX/VMS; UTX/32S; MPE V/E; PRIME-OS; Top Secret(MVS); Unisys A-Series; CONCURRENT OS-32; EXEC-1100; WANG SVS/OS

Access by Regulation (Law Clearance):
AT&T SV/MLS; UNISYS OS-100; MULTICS; VSLAN; RACF (MVS-ESA); TRUSTED XENIX; DEC CMW; HARRIS CMW; HP UNIX WS; XTS-200; SECUREWARE CMW;SUN CMW

Verified Protection:
SCOMP; GEMSOS; Multinet Gateway; MLS LAN; LOCK

Figure 2

Not Evaluated Security Mechanisms

a sampler

ACSI COMPSEC II
AST Knight DSM
CSC Sentinel
MPPi PC Lock III
OkioK RAC/M
Programit Savant Guard
Sophco PROTEC
Western ONGUARD
Winterhalter SECURE
Tower Systems Surveillance

NETGUARD
Tower System BSAFE!
Unitec USecure
VM Software VMSecure
Timelock TIMELOCK
Computer Synergetics KEYSTONE
Goal Systems ALERT
Legist Automation MS/CS
SATCOM Interface/3000
Security Lockit

Figure 3

DAC MADE EASY?

One common pitfall which seems to defy avoidance, at least in security literature, is the selection of access control software to comply with C2 policy. The work-horse of government computing architectures over the years has been the IBM System 370. Several products have been evaluated and found to provide C2 level of protection with various System 370 operating systems. Some other COTS products provide most of C2 but have not been evaluated by any independent entity. So sifting through the marketing glossies, attending product briefs, etc, trying even to design a system with any level of protection can be a monumental effort.

METHOD OR MADNESS?

No recognized heuristic exists for selecting from among the various products. Some common reasons given for choosing a particular product have been:

- We've used this product before and we like it
- This product is developed by the developers of our operating system and we're more comfortable with a single vendor
- That company has been a take-over target lately and we're afraid to use their product
- This product is more user-friendly than those others
- This product won't be so easy to figure out for the average hacker, than those user friendly ones
- The boss has used this product before and he likes it

To that list, recent analysis by an experienced security integrator would also bring into consideration:

- **management's inclination towards centralized administration**. Some products lean towards centralized control and some are more efficient if control is delegated.
- **distributed security operation**. If the operational needs dictate that security administration tasks such as management of userids and passwords continue without reliance on a remote central site.
- **user dataset activity.** If a typical user accesses numerous datasets to get a job done then a scheme which checks a user profile instead of a dataset Access Control List (ACL) may be more efficient over a large number of users.
- **dataset volatility.** If system datasets have short lives, that is get deleted and created frequently, then a scheme which uses profiles may be cumbersome because the datasets would have to be deleted from each of the affected user profiles.
- **user id velocity.** In systems which have a large number of users or in which users come and go frequently (such as a widely dispersed C3 network), then dataset ACLs become cumbersome to manage and user profiles might offer more palatable operations scenarios.
- **compatibility with other security mechanisms.** Several special purpose identification devices, such as retina scanners, smart card readers, etc are available to use in concert with access control software. If the system architecture depends

on using such a device, the access software must have appropriate hooks to allow the implementation to include the device.

WHERE'S THE SECURITY BARRIER?

Making informed decisions, even at the entry level to security, requires an in-depth analysis of the operation of all the applications on a system, as well as the operating system, the security administration organization, and the access control package itself. A DIY-er will frequently founder before getting a good start because of the sheer immenseness of the analysis. The services of an integrator to do all this analysis early in the planning process can prevent numerous headaches later.

One measure of the complexity of the problem of trying to implement security in large System 370 ADP sites is system audit results. Last year the President's Council on Integrity and Efficiency (PCIE) Computer Systems Integrity Project reviewed nine computer centers in the Federal government[1]. Each of them was operating a System 370 using the MVS operating system and one of the commercially available security packages. The overall condition described by the Inspectors General included the existence of serious control deficiencies at all agency computer centers reviewed.

Conclusion: putting a C2 package on a system can fall significantly short of meeting the system security requirements of C2.

The analysis done at one Sys-370 installation, supporting a major DoD contract, found fifteen features and facilities of the operating system which must be tightly controlled in order to give any degree of assurance that the system could be certified at any TCSEC level of security:

Authorized Program Facility (APF)
Program Properties Table (PPT)
Ext SVC Routing Facility (SVC/ESR)
Input/Output (I/O) Appendages
System Started Task Control
MPF and Automated Operations
User Attributes Data Set (UADS)
System Modification Program (SMP)
Supervisor Calls (SVC)
LinkList
LPAList
MVS Exits
Hooks to MVS
Catalogs
Subsystems

Only about one-fourth of the installed software required the above special privileges to operate, and therefore was considered part of the Trusted Computing Base (TCB). Any TCB software must be thoroughly examined to ensure that system privileges are not abused. About one-fourth of the installed software required no special privilege or facility and was thus considered to be non-security relevant.

The remaining 50% was judged security relevant (non-TCB) software. The reason for this inordinate amount of security relevancy is that the operating system provides a tremendous flexibility to applications through the use of these special features and facilities. Developers use this flexibility to enhance the performance of their products, but usually without regard to the security impacts of their actions.

The foregoing discussion clearly indicates that C2 is not trivial to do. Higher levels of security are even more complex. In the TCSEC, a B2 system has eleven more security policy requirements than a C2. A B2 also has four more accountability requirements, nine more assurance requirements and five more documentation requirements than a C2. Each of these requirements must either be demonstrated through testing or delivered as a document.

WHAT ABOUT MAC?

The NCSC's Environments Guidelines advise that B2 Mandatory Access Protection is the lowest level of security functionality at which a system may be trusted to separate uncleared users from classified data. Although a system provides mandatory labels at B1, the design assurances aren't sufficient to trust the system to protect classified data.

The most common uses for Mandatory Protection products are as trusted guards, allowing information to flow upward from one environment to an adjacent one while preventing downward flow. The level of complexity in the analysis of a C2 add-on package pales in comparison with that required to implement two C2 systems connected by a B2 guard.

WORK BREAKDOWN STRUCTURE

An early decision to be made in any secure system design is how the architecture is to be arranged to accomplish the security mission. What I mean here is that when cooperating components are to be relied upon to work together to achieve some function, there must be a fairly clear division of labor. Security is generally thought of in terms of four functions:

- Identification (usually coupled with Authentication)
- Audit
- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC).

I have briefly dealt with the last two, namely that DAC is control by the "owner" of the data and that MAC is control by clearance, classification or some other law or regulation. MAC is not optional it is defined by higher authority.

Identification (Id) and Authentication are the fundamental mechanisms by which the system knows who a user is. They so frequently occur together that we tend to link them in our thinking. An example of Id alone is when game software asks for names for player number 1, number 2, etc. There isn't any authentication because the system only uses the names to keep the scores correctly. Secure systems, even at the C1 level, require that users also be Authenticated by some protected mechanism.

Authentication most typically is accomplished by having the user supply a password, which, when coupled with the name (userid) supplied during Id, gives the system information which can be compared to its own database. Some devices such as retina scanners, smartcards, etc also are used as Authentication mechanisms, sometimes in addition to a password.

Audit is the requirement for the system to keep a private record of what is going on in the system. Audit is first required at C2. Simply stated, it requires that the TCB must have the capability to record uses (or attempted use) of the Id mechanism, creation and deletion of data items, and other security relevant events. This includes actions of any "trusted" user such as security administrators, audit administrators, and computer operators. The audit log can then be used to review the day's operations, and with some automated help, assist in detecting aberrant behavior.

COOPERATIVE CONNECTIONS

Having thus divided the security function into some more or less independent subfunctions, the system designer can consider how the components of the system will cooperate to accomplish the function.

One way to connect them together is in Distributed fashion (Figure 4). In this arrangement the functionality is well segmented and each component'relies on the proper operation of the others. One big advantage of this arrangement is ease of evaluation. Most early Trusted systems used an arrangement like this, although the whole diagram depicted a single computer operating system with the subfunctions being performed by program modules.

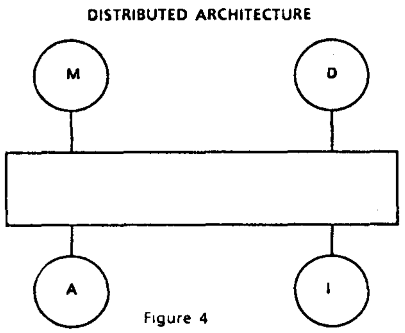


Figure 4

A specific challenge of this architecture, when the components are more than a few inches apart, is the communication medium. This can be thought through by considering a bus with several slots in it. In one slot is a DAC module on a board, one is a Mac module, etc.

The overall operation of the architecture is then dependent on the speed and reliability of the bus. A request for access to some data by some user might require six or more high priority messages across the bus to accomplish the access. Another arrangement of cooperating components is termed Layered (Figure 5). In this approach each layer has responsibility for a major security function (MAC, DAC or Id) and the audit associated with that function. In this arrangement a user's request for access is passed along from one layer to the next, but the audit transactions are filed locally within the layer. This architecture is most advantageous when the layers are physical components in a chain or on a network. The major challenges are in ensuring that the coded clearance levels and classification levels are consistent, and in trying to consolidate the various audit trails to make some sense of the data.

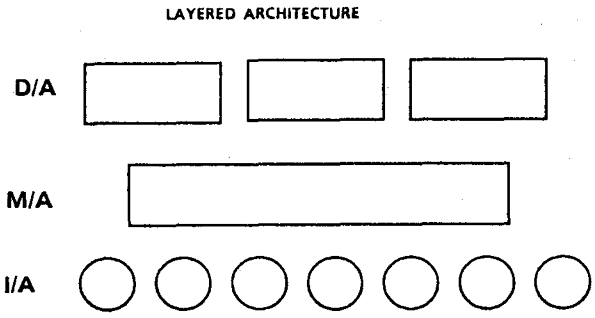


Figure 5

MIX 'N' MATCH

There are, of course, many variations of arranging security components. In the example previously mentioned where a B2 gateway connected two C2 systems, one could program the B2 component to only perform MAC and Audit, or indeed provide vide a full set of B2 capabilities. One example would be to allow only selected (dis-cretionary) users to pass data through the guard. Audit messages could be filed within the guard, or passed back to the systems which initiated the actions being audited.

To complicate the world just a little more, the products on the Evaluated Products List (EPL) encompass several architectural flavors:

- Systems designed with security built-in (SCOMP, MULTICS, VAX/VMS etc)
- Security add-on Subsystems (LOCK, Trusted Xenix, RACF etc)
- Security Enhancing Components (Netguard, Safeword, Gordian etc)

These products may be combined as layered or distributed systems as described above, but divining the protection provided by the resultant system would surely be beyond the scope or capability of most integrators, much less a DIY-er.

CONCLUSION

Architecture decisions have significant impact on system life-cycle costs. During development and implementation these costs appear as direct costs of developing "trusted" program modules, devising test scenarios which demonstrate that the system enforces the security policy, and writing (or rewriting) documentation which supports an accreditation decision. During operation, living with the architecture decision may impose significant constraints.

Changing "trusted" programs is extremely cumbersome, increased usage of system resources may cause operational decisions to circumvent security features, and high levels of frustration may be blamed on security, when in fact a design decision is the trouble.

A commander faced with controlled dissemination requirements, and multilevel security operations, can meet mission needs most effectively by having requirements analyzed by a security integrator. Very few teams have successfully implemented security solutions using commercial products without the integrator. The probability of success in a project such as the design of a new C3I system will be greatly increased if the team has struggled with a similar problem before. A background of experience and a systematic approach to security problem solving will pay big dividends in user satisfaction, and management peace of mind.

Reference

1. PCIE Review of General Controls in Federal Computer Systems, Oct 88

A Taxonomy of Security-Relevant Knowledge

Gary W. Smith

National Defense University
Information Resources Management College
Wahington Navy Yard, Bldg 175
Washington, DC 20374

Abstract. Developing a multilevel secure automated information system requires a significant amount of data, information, and knowledge about the application domain to be supported. Much of that knowledge is independent of the security mode of operation (e.g., system high or multilevel). On the other hand, effective (that is, usable by and acceptable to the user) multilevel secure application systems will place significant demands on the application designer. Certainly, the security requirements will be more than specifying the highest classification of data and the lowest user clearance. Before one can hope to develop an automated system that meets the user's security needs, all the security-relevant knowledge for the application domain must be identified. This paper provides a taxonomy of security-relevant knowledge that must be considered when specifying and designing a multilevel secure application system.

1 Introduction

When one develops an automation information system (AIS), a large quantity of knowledge about the application domain must be incorporated into the system design and specification. Adding multilevel security as a system requirement introduces additional complexity and challenges for application developers. It is essential that user's secrecy requirements for the data maintained by the system, behavior and operation of the system be identified at the beginning of the system life cycle. These requirements drive the application system and database design -- all of which are not unique to trusted systems. An additional challenge is added when one considers multilevel security requirements. Specifically, the system must be certified to meet the required level of trust in accordance with the Trusted Computer Security Evaluation Criteria (TCSEC) [1]. Effective multilevel application systems will need more trusted functionality than just an operating system. One could hope that multilevel database systems will provide significantly greater functionality -- and will have the required level of trust. Unfortunately, we believe that users will demand fully functional, multilevel secure systems, and those systems will require application dependent functionality which is "trusted." For application system and database designers to meet this challenge, additional information must be explicitly identified and considered during the design and specification of the system -- information we term *security-relevant knowledge*.

Security-Relevant Knowledge is about the application domain -- its users, data, rules, procedures, and other security requirements. In essence, it provides the detailed security requirements and related information. As such, security-relevant knowledge is not about, nor limited by, technology. For example, nowhere will you see anything that resembles the *-property of the Bell-LaPadula Security Policy Model [2]. There is a simple explanation for this apparent omission: the *-property is not a requirement of any application domain; there is no parallel in the manual world since we automatically "trust" humans to properly classify and downgrade information or data. The *-property is a direct result of the limitations of technology; if we could "trust" all software, then Trojan horses would not be a problem and "writing down" in classification level would be acceptable.

The remainder of the paper presents a taxonomy of the two classes of Security-Relevant Knowledge, external and application-specific. *External knowledge* consists of those rules and facts that are dictated by sources outside the application domain. Two categories of external knowledge, policies and facts are described in Section 2. *Application-specific knowledge* consists of rules and facts that are applicable only to the application domain. Six categories of application-specific knowledge are identified in Section 3: policies, data integrity semantics, data secrecy semantics, user profiles, access control requirements, and operational/managerial requirements. An overview of the taxonomies of data integrity and secrecy semantics (Sections 3.2 and 3.3) was previously presented [3]. This paper extends that work by providing a detailed explanation and examples for each type of constraint included those taxonomies.

The word *policy* is used in this paper with a less restrictive meaning than normally found in the computer security community. In particular, a policy is a statement of a guiding priniciple or procedure. A policy, used in this context, need not be (but is not precluded from being) something that is enforced by a trusted computing base (TCB). One of the challenges of implementing usable multilevel systems is determining how much of the security-relevant knowledge (i.e., security requirements) must be included in the security policy and therefore enforced by the TCB.

A simple application domain is used to illustrate each category of knowledge. The application domain is a portion of the "corporate database" for a large, hypothetical defense contractor who specializes in designing and implementing automated intelligence systems. There are three entities of interest in the corporate database--*employees*, *clients* and *projects*. Figure 1 is graphical portrayal of the corporate database, where circles represent entities and rectangles are attributes of entities. Attributes of entities are referenced using the "dot notation" (e.g., employee.ID).

The employee entity represents persons employed by the corporation; it has six attributes: an employee number (employee.ID), a name (employee.name), a birthday (employee.birthday), a salary (employee.salary), a department to which the employee is assigned (employee.dept), and one or more skill codes (employee.skill).

The project entity represents projects on which the corporation is working and has five attributes: a project number (project.ID), a name (project.name), a cover story subject

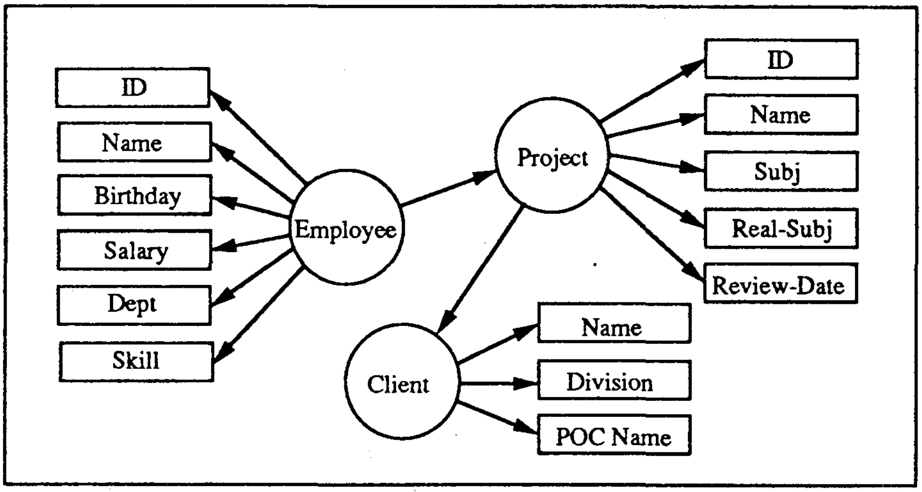


Figure 1 - Corporate Database Schema

(project.subj) for the actual subject of the project (project.real-subj), and a date for the next project review (project.review-date).

The client entity represents the clients that are the sponsors of projects. The client entity has three attributes: the company name (client.name), the major organizational element sponsoring the project (client.division) and the name of the point of contact for the client (client.POC-name).

2 External Knowledge

External knowledge consists of those rules and facts which are dictated by sources outside the application domain. Sources for external knowledge include local and national laws, the corporate headquarters, and other external elements. There are two types of external knowledge -- policies and facts.

2.1 External Policies

External policies are the general rules that apply to the specific application domain as well as other application domains. Such policies are statements of general requirements to which the application system must comply. Note that "policy" as used in this context has a broader meaning than the restrictive definition of the TCSEC (i.e., security policies are those policies enforced by the TCB.) Required policies are generated by several kinds of external sources. Example include:

National or local laws:

- Sensitive defense information must be protected in accordance with national security directives.
- A minimum of one-tenth of an employee's pay must be withheld each month and sent to the State Treasurer.
- Records of sales for item X must be kept for 10 years.

Corporate policies and regulations:

- Avoid the over classification of data.
- All projects where the Department of Defense is the client will be assigned a need-to-know category.
- Only the corporate security officer can give security clearances to employees.

Lower-level policies which may affect only one or few divisions of the corporation:

- For the Engineering Department, an automated system must be able to reconstruct who did what to whom for the past two years.
- All research and development information will be protected as sensitive information.
- Reports from selected departments must be approved for release by the appropriate department head.

2.2. External Facts

The second type of external knowledge consists of *facts* about the environment in which the application exists. External facts are important since they must be explicitly considered when assigning a classification level to the data in the application domain. External facts are normally general knowledge and sometimes may even be just common sense. Examples include:

- The corporation supports the intelligence community.
- Everyone knows the corporation supports the intelligence community.
- The Department of Defense is the client for project XYZ, and project XYZ is protected by need-to-know category A.

3 Application-Specific Knowledge

Application designers and security officers have little control over the external knowledge; however, they can influence *application-specific* knowledge. In a manner similar to external knowledge, application specific knowledge consists of rules and facts that only affect the security of the application domain. The potential volume and complexity of application-specific knowledge is, however, much greater than external knowledge. Application-specific knowledge is divided into six categories: policies, data integrity semantics, data secrecy semantics, user profiles, access control requirements, and operational/managerial requirements. Each category of application specific knowledge will be discussed in turn.

3.1 Policies.

Application-specific policies are applicable only to the application domain in question. They must be consistent with and supportive of higher-level external policies. One type of policy specifies proper usage of the system. Examples include:

- Users authorized to update project data for sensitive projects will be limited to the minimum number possible to accomplish the mission. (Note that this policy is general in nature and, if appropriate, could be applied to more than one application domain.)
- Information about Project XYZ is classified S in this application domain, even though in other domains it is only C.

Other policies may state detail rules for complying with general corporate policies contained in the external knowledge. For example:

- All accesses to project.real-subj will be audited.
- Audit trails for all updates to project data will be retained for 2 years.

Still other policies reflect how the system should respond to unauthorized behavior by a user. Examples include:

- For any query in which employee.salary is among the attributes requested but the user is not authorized to access employee.salary, the authorized attributes will be provided to the user and the event recorded in the audit.
- For any query that includes data on any classified project for which the user is not authorized, the following actions will be taken: the entire query will cancelled, the user will be given the message "improper query," a message will be sent to the system security officer with the copy of the query and reason for rejection, and the event will be recorded in the audit.
- Any access to project.name = ISTAR will result in locking the user's terminal and ringing the bell on the system security officer's terminal with a message stating the security violation, room number and name of the user.

3.2 Data Integrity Semantics

The data integrity semantics of an application domain are expressed using integrity constraints. *Data Integrity constraints* are the traditional "update semantics" of the database community -- constraints which specify the valid relationships between the data and the rules for ensuring the validity of the data.

Integrity constraints can be stated for data objects (e.g., entities and attributes) and relationships between data objects (e.g., relationships between two or more entities or relationships between entities and their attributes). The following paragraphs provide a detailed taxonomy of the types of data integrity constraints for entities, attributes and relationships.

(1) **Data Integrity Constraints on Entities.** The following are the types of data integrity constraints that can be stated about entities. For each type of data integrity constraint one or more examples are given.

Uniqueness - by definition, each instance of an entity must be unique.

- Each instance of the employee entity is unique.

Minimum Cardinality - the minimum number of instances that must exist in the database.

- There must be at least one instance of the employee entity.

Maximum Cardinality - the maximum number of instances of an entity allowed in the database.

- There can be no more than 25 instances of the project entity in the database.

Key(s) - attribute(s) whose values uniquely identify each instance of the entity. In some cases it may take the combination of two or more attributes to form a key. In other cases there may be more than one attribute that qualifies as a key, one of which is designated the primary key.

- Both project.ID and project.name are keys for the project entity.
- The key for the client entity is the combination of two attributes--the company name (client.name) and the major subdivision of the company (client.division).

Key Value-Type - the valid types of data for the entity key. The traditional value-types are Real, Integer, Boolean, and String. In order to capture security-relevant knowledge about an application, the value-type String must be extended to indicate when the data values contain semantic information.

A-String - of type String where the value reveals All semantic information.

- Plain text fields such as employee.name or project.subj.
- Free-form text fields such as "remarks."

P-String - of type String where the value may reveal Partial information.

- A attribute which contains coded data and where some instances of the data may reveal some information. For example, ENG312, ENG515, ORSA960 are values for employee.skill which provide information: the alpha characters represent the basic skill group -- ENG means engineering and ORSA means operations research and systems analysis; the first character of the three numeric characters represents the skill level -- 3 is entry level, 5 is mid level, and 9 is senior level; and the last two digits of the three numeric characters represent a sub-skill -- 12 is electrical engineering, 15 is mechanical engineering, and 60 is stochastic modeling.

N-String - of type String where the value contains No semantic information.

- Values for project.ID which are random but unique.
- Values which have been encrypted.

Key Value-Set - the set of valid values for an attribute which is a key. The value-set can be of two types: a range of values, an enumerated set of values, or a named set of values based upon specific criteria. (The latter set is analogous to the notion of strong typing in programming languages such as Pascal and Ada.)

- Range: project.ID must an integer between 100 and 999.
- Enumerated Set: project.name must be one of the following -- ISTAR, GEMINI, EPCOT, ALPHA, BARBER or CALS.
- Named Set: project.review-date is of type *date* which is a data type where the values meet the following criteria: the first two characters must be 1-12; the second and third characters must be between 1 and 31 if the first two characters are one of the following 01, 03, 05 07, 08, 10, or 12; etc.

Key Size - a measure of the maximum size of the field containing the key (normally the number of characters).

- Employee.ID has a maximum size of nine characters.

(2) **Data Integrity Constraints on Attributes.** One can speak of three types of data integrity constraints for attributes: value-type and value-set, and transition compatibility. The first two types of constraints are exactly the same notions as defined above for keys for an entity and need not be discussed further. The third type is:

Transition Compatibility - when the value for the attribute changes, the new value must satisfy a specified relationship with the old value.

- New.date must always be greater than old.date.

Other types of constraints that often are applied to attributes are actually constraints on a relationship between an entity and one of its attributes.

(3) **Types of Relationships.** Before discussing the types of data integrity constraints applicable to relationships, it is necessary to present basic types of relationships. Figure 2 displays the fundamental types of relationships among combinations of entities (circles) and attributes (rectangles).

Figures 2a through 2f and 2h are the basic relationships. Figure 2a shows the relationship between an entity and one of its attributes. Figure 2b shows the relationship between two entities. A relationship between two attributes of the same entity is shown in Figure 2c. Figure 2d shows the relationship between two instances of the same entity and Figure 2e shows the same relationship between two instances of an attribute. Figure 2f shows a relationship between an entity and an attribute of a different entity. Figure 2h shows the same type of relationship but between attributes of different entities.

Figures 2g, and 2i through l are fundamental combinations of the basic relationships. Figure 2g is similar to 2f, differing only by having a basic relationship between the two entities. In a similar manner, Figures 2h and 2i differ only by whether there is a relationship between the two entities. The remaining examples are of more complex combinations of basic relationships: Figure 2j shows a cycle, Figure 2k shows multiple paths between objects, and Figure 2l represents combinations which do not contain a cycle or multiple paths.

The following notation will be used: a Type *X* relationship will refer to the relationship shown in Figure 2*x*.

(4) **Data Integrity Constraints on Relationships.** It is possible to state a number of different types of data integrity constraints for relationships between data object A and data object B.

Minimum Cardinality - the minimum number of instances of B required for each instance of A. The constraint can be uniformly applied to all instances of A, or may be based some conditions (e.g., the value of A).

- uniform: each employee (A) must be assigned to at least one project (B).
- conditional: all employees in the Engineering Department must be assigned to at least two projects.

A common integrity constraint states that for each instance of an entity an instance of an another attribute (or entity) is "required." This required constraint is represented using a minimum cardinality of at least one. An entity (or attribute) that is not required would have a minimum cardinality of zero.

Maximum Cardinality - the maximum number of instances of B allowed for each instance of A. This is similar to Minimum Cardinality in that the constraint may be uniform or conditional.

- uniform: each employee (A) can have a maximum of one name (B).
- conditional: each employee in the Engineering Department can have a maximum of six skill-codes.

Uniqueness - for each instance of A there must be a unique instance of B. In this case, Minimum Cardinality equals Maximum Cardinality which equals one.

- Each employee has a unique ID.

Internal Compatibility - the values of A must meet a specified relationship with the values of B. The relationships can be uniformly or conditionally applied to both A and/or B.

uniform A - uniform B: for all instances of A, B must have a fixed relationship with A.

- The President's salary must be greater than all other instances of employee salary. (This is an example of a Type E relationship from Figure 2.)

uniform A - conditional B: for all instances of A, B must meet a set of conditions. (This type of compatibility is logically equivalent to value-type and/or value-set constraints on object B, but is included for completeness.)

- For all instances of Project, Review-Date must be of type "date."

conditional A - uniform B: for all instances of A that meet a set of conditions, B must have a fixed relationship with A.

- All employees assigned to the Quality Assurance Department must have the skill-code QA752.

conditional A - conditional B: for instances of A that meet a set of conditions, B must be meet a set of conditions.

- All employees assigned to the ISTAR project must have one of the following skill-codes: ENG612, ENG315, or ENG632.

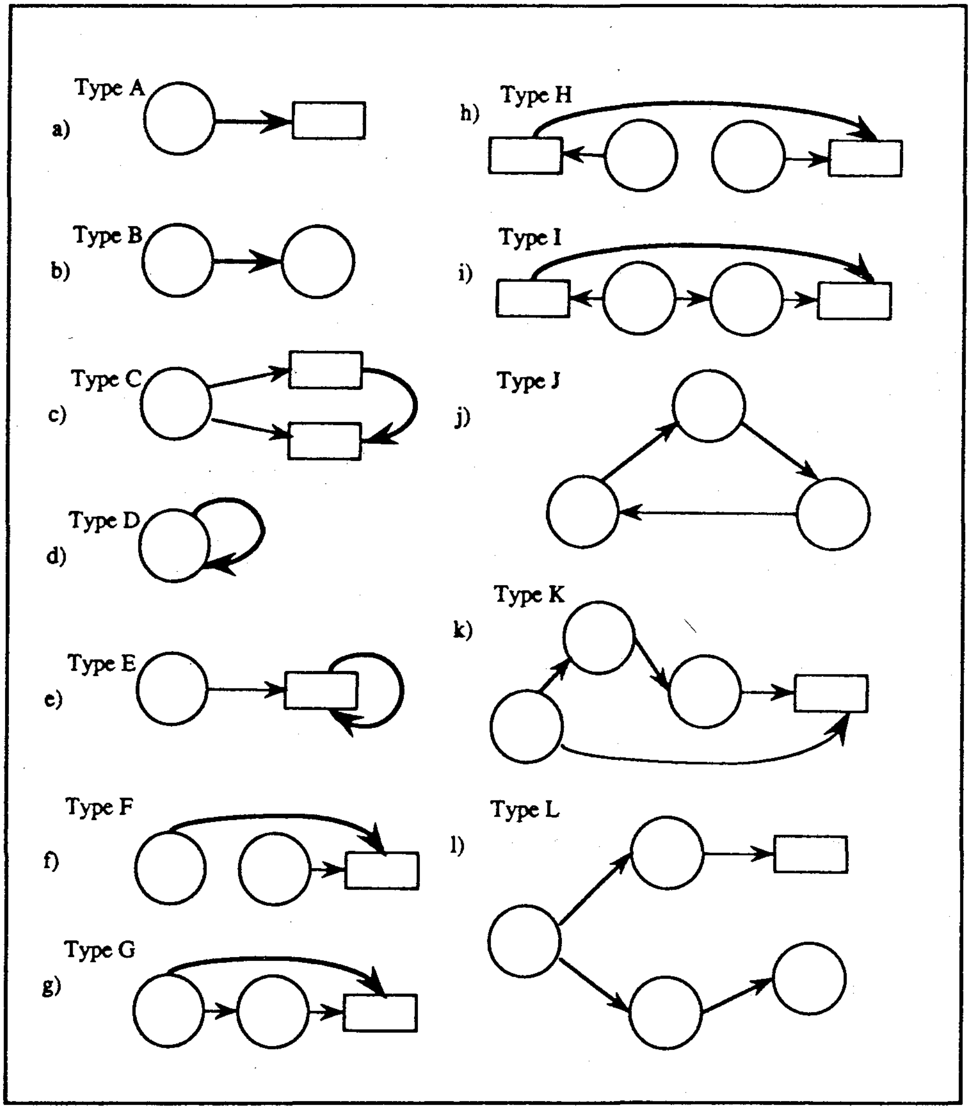


Figure 2 - Types of Relationships

External Compatibility - the values of A must meet a specified relationship with the values of B based upon some criteria involving an external (i.e., not A or B) constant or data object.

Uniform: for all instances of A and B, the relationship between A and B are constrained by object C.

- The sum of all instances of employee.salary must be less than $1M (a constant) or the value of a notional attribute called budget.total
- The sum of A and B is less than C.

Conditional: instances of the relationship between A and B are constrained if some condition for the value of C is met.

- If the value of C is less than 100, then A must be less than B.

Both internal and external compatibility constraints illustrated above are basic types of constraints. They can be combined to from more complex constraints.

- If A is less than 100, B is less than 200, and C is less than 300, then D must be greater E.

3.3 Data Secrecy Semantics

The taxonomy of data secrecy semantics parallels that of the data integrity semantics taxonomy. Secrecy semantics involve rules called *data secrecy constraints* [3] by which data and combinations of data must be classified. The assignment of a classification level within these rules must take into consideration all external knowledge--policies and facts. One can state data secrecy constraints for entities, attributes, and relationships.

(1) Data Secrecy Constraints on Entities. The following types of secrecy constraints can be stated for entities.

Classification of an Instance - the classification of each instance of the entity is based upon one or more of the following criteria:

Internal Criteria. Classification is based upon one or more of the attributes of the entity.

Uniform - all instances have the same classification.

- all projects are classified TS.

Conditional (Range) - classification of an instance is based upon a range of values.

- all projects with a project ID of greater 500 are TS.

Conditional (Set) - classification is based upon an enumerated set of values.

- projects with the following names are TS: ISTAR, GEMINI, and EPCOT.

User specified - the user establishes the classification at insertion time.

- the classification for projects will be specified by the user who creates the project entity at the time the instance is created.

External Criteria. The classification is based upon criteria not a part of the entity.

- Projects are classified TS if the Client is Company X.
- Projects are classified TS if it is Tuesday.

Classification of an Entity Name - the name of an entity can be classified. Only uniform classification is allowed.

- The name of the project entity is classified S.

Cardinality Aggregation - the aggregation of N instances of an entity are unclassified, but N + 1 instances are classified.

- The aggregation of more than 25 employees is classified S.

Hiding the Existence of Instances - in some operational environments, it is necessary to hide the fact that classified instances of data are in the database. The existence of data may need to be hidden based upon one or more of the following criteria: uniform for all instances, conditional based upon the instance, and conditional based upon the classification level.

- uniform: hide the existence of all projects.

- conditional (instance): hide the existence of the ISTAR and GEMINI projects.
- conditional (classification): hide the existence of all projects classified TS.

Note that this category of knowledge involves a requirement to hide the existenceof data based on the needs of the application domain. As oppposed to the need to hide the existence of data from Trojan horses that may be receiving data through a signalling channel.

Identificate(s) - the set of all identificates of an entity. An important notion, which is fundamentally new, is that of an identificate. An *identificate* is defined as an attribute of an entity which, from a secrecy perspective, allows the object to be identified. In traditional database terminology, we have just described a key -- a data element (or combination of data elements) which "uniquely" defines the object. However, when considering secrecy, a unique attribute (i.e. a key) is not sufficient. One must also be concerned with an attribute which is "nearly-unique," that is, a *near-key*. A near-key is an attribute that will provide the identity of the object the majority of the time. Thus an identificate is an attribute that is either a key or near-key.

- The identificates for the Employee entity are ID and Name.

(2) Data Secrecy on Raw Data - although raw data (e.g., a string of characters which are not related at the moment to any metadata) are rarely classified, it is included for completeness.

- The word (or string of characters) GEMINI is classified TS.

(3) Data Secrecy Constraints on Attributes - three types of data secrecy constraints can be stated for attributes: classification of an instance, classification of an attribute name, hiding the existence of instances. Since these types of constraints are analogous to data secrecy constraints on entities, detailed examples are not given.

(4) Data Secrecy Constraints on Relationships - the types of relationships for which data secrecy constraints can be stated are the same as those described in Section 3.2.2. All types of data secrecy constraints are applicable to each type of relationship. The following paragraphs describe different types of data secrecy constraints that can be stated for relationships between entity/attribute A and entity/attribute B.

Classification of Relationships - the association of two data objects (entities or attributes) is classified at a level independent of the classification of the two data objects. In a manner similar to data integrity compatibility constraints, the classification of the relationship is based upon either internal criteria (i.e., conditions involving the two objects) or external criteria (conditions not involving the two objects).

Internal Criteria. This criteria may be a uniform or conditional selection based upon the value of instances for both data objects A and B.

- uniform A - uniform B: the classification of the association of the employee entity (A) and employee.salary (B) is S.
- uniform A - conditional B: the classification of the association of the employee entity and employee.salary is S when employee.salary is greater than 100K.
- conditional A - uniform B: the classification of the association of the employee entity and employee.salary is S when the employee's department is Executive Office.
- conditional A - conditional B: the classification of the association of the employee entity and employee.salary is S when the employee's department is Executive Office or salary is greater than 100K.

External Criteria. Classification of the relationship is based on some external criteria not involving objects A and B.

- the classification of the association of the employee entity with employee.salary is classified S when the employee is assigned to project ISTAR.

Hiding the Existence of a Relationship - the existence of a relationship between two data objects may need to be hidden from unauthorized users. The requirment to hide the existence of a relationship is based upon one or more of the following conditions: uniform for all instances, conditional on the value of instances of A or B, conditional on the classification of A or B.

- uniform: hide the existence of all instances of the relationship between employee entity and project entity.
- conditional (instances): hide the existence of instances of the relationship between employee entity and project entity when the project name is ISTAR or GEMINI.
- conditional (classification): hide the existence of instances of the relationship between employee entity and project entity when the project is classified TS.

(5) **Data Secrecy Constraints on Constraints** - data integrity and/or secrecy constraints may contain classified information and therefore may also be classified.

- The integrity constraint "project.name must be ISTAR, GEMINI, etc." would be classified S if the fact that "ISTAR is the name of a project" is classified S.
- The secrecy constraint "the attribute name, project.real-subj, is classified TS" (which might be required to hide the existence of a cover story) would be classified TS.

3.4 User Profile.

In order to perform access control functions, knowledge about users and groups of users is required. The most basic piece of knowledge is a user's clearance. Each user must have a trust level (hierarchical) and can have one or more formal need-to-know categories. Users may be included in a group of users. Groups may be based on organizational structure, roles, or may be *ad hoc*.

- organization structure: all users assigned to the personnel officer are a group called *personnel office*.
- role: all "personnel clerks" or "personnel authorized to validate expenditure of funds."
- *ad hoc*: only Smith and Jones are authorized to perform action X.

In the context of government, nationality may also be a relevant property of a user.

3.5 Access Control Requirements.

One of the primary aspects of security, and the one that has received the most attention in the past, is access control -- mediating which users can access (for read, write, or read/write) each data object. There are two dimensions to access control. The first dimension is determining who is authorized access; the second dimension is what type of access is authorized. For each constraint, the specific type of access (read, write, read/write) must be specified. The following types of access control constraints can be stated for an application domain.

(1) **Access Control Dimension - Who.** The first consideration is determining if a user is authorized to access data. The first constraint is the simple security property which is mandatory access control (MAC) as presented in the Orange Book where access control is based upon sensitivity labels assigned to data objects. All other types of constraints implement discretionary-like requirements for need-to-know. Using sensitivity labels to enforce these additional types of constraints is normally not feasible.

Simple Security Property. The clearance of the user must dominate the classification of the data object. This is the overriding access control constraint. All other constraints can be applied only after meeting this constraint.

Property of the User. Access is based upon some property of the user (other than clearance). Properties may be the result of data explicitly represented in the database:

- only personnel assigned to the Personnel Department can access object X.
- only personnel located at building A can access object X.

- only personnel clerks (i.e., job-title is personnel clerk) can access object X.

Properties of a user may also be part of a user profile:

- only members of XYZ group can access object X.
- only US Citizens can access object X.

Properties of a user may also be implicit in the organization structure or management responsibilities:

- only supervisors in the Purchasing Department can authorize purchases.
- managers can only see data on the employees they supervise.

Property of the Data. This type of constraints involves restricting access based upon the data previously accessed. They can be used to specify unique secrecy policies such the Chinese Wall policy [4]. The constraints must be stated relative to data objects (or instances of a data objects) A and B. The constraints can be uniformly or conditionally applied to either data object A or B. Each constraint may only be applicable for a specified period of time (e.g., for a period of one year).

- uniform A - uniform B: If a user has seen client data, then he/she is not authorized to see data on any project.
- uniform A - conditional B: If a user has seen client data, then he/she is not authorized to see data on the ISTAR project.
- conditional A - uniform B: If a user has seen data on the ISTAR project then he/she is not authorized to see data on any other project.
- conditional A - conditional B: if a user has seen data on the ISTAR project, then he/she is not authorized to see data on the GEMINI project.

Cardinality of Instances. Access to a data object is restricted based upon the number of instances of the data object which can be accessed.

- Smith is allowed to read only a maximum of 10 employee records.

Combinations of Properties. Access can be restricted based upon a combination of two or more of the properties described above.

- Only members of Group A can see Object X if they are US Citizens and have not seen instances of Object Y.

(2) **Access Control Dimension - Type.** The dimension of access control is to what type of access the user is authorized.

Viewers of Instances. Access is restricted to read an instance of a data object. These examples show how this type of access can be applied either uniformly or conditionally to a data object.

- uniform: Only personnel clerks can view employee salary.
- conditional: Only the project manager can see salary for employees assigned to their project.

Modifiers of Instances. Access to modify an instance of a data object is authorized. In the context of a database that contains structured data (as opposed to text) the term "modify" can mean three types of changes to data: adding (or inserting) new records (tuples), deleting records (tuples) and modifying the values of existing records (tuples). A robust security policy requires that access control be adjudicated for each type of change. Therefore, three types of change are explicitly described: modification, creation, and deletion. Modification is defined to be any changes of values for instances of a data object. This type of constraint can be applied either uniformly or conditionally to a data object.

- uniform: Only personnel clerks can modify instances of the employee entity.
- conditional: Only Smith can modify instances of the employee entity assigned to the Engineering Department.

Creators of Instances. Access to create an instance of a data object is authorized. This type of constraint can be applied either uniformly or conditionally to a data object.

- uniform: Only personnel clerks can create instances of the employee entity.
- conditional: Only Smith can create instances of the employee entity assigned to the Engineering Department.

Deletors of Instances. Access to delete an instance of a data object is authorized. This type of constraint can be applied either uniformly or conditionally to a data object.

- uniform: Only personnel clerks can delete instances of the employee entity.
- conditional: Only Smith can delete instances of the employee entity assigned to the Engineering Department.

3.6 Operational and Management Requirements

There are many security relevant aspects involving the operation and management of the application system which must be specified. These considerations are dynamic in that they involve constraints on actions taken by users or the system over time. Examples include:

- (shared responsibility) There must two person control for updating critical data.
- (personal approval) Managers must approve the assignment of a classification level to certain data.
- (data availability) Standard personnel reports must be available by 8a. each work day.
- (change timeliness) Changes in user profiles must take affect within five minutes of the change.
- (system behavior) If any unauthorized user attempts to access any information in the ISTAR, immediately notify the Security Officer.

4 Conclusions

Implementing fully functional, multilevel secure automated application systems is a significant challenge. The taxonomy presented here is comprehensive list of the security-relevant knowledge that must be explicitly identified early in the life cycle of the automated system. The design and specification for the system will derive from this knowledge.

The data integrity and secrecy semantics must be addressed at this level of detail to support analysis of the requirements and database design for inference and signaling vulnerabilities.

An additional challenge is to establish system security requirements at the appropriate granularity. For example, how the system responds to an attempted unauthorized access to one database (e.g., audit the attempt) may differ greatly from another database (e.g., notify the security officer).

References

[1] DOD-STD 5200-28, *Trusted Computer System Evaluation Critiria*, 1985.

[2] Bell, D. E. and LaPadula, L. J., *Secure Computer Systems: Mathematical Foundations*, Vols I-III, Mitre Corporation Technical Report, November, 1974-June, 1974.

[3] Smith,G. W., Modeling Security-Relevant Data Semantics, *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*. pp. 384-391.

[4] Brewer, D. F. C. and Nash, M. J., The Chinesse Wall Security Policy, *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, May, 1989, pp. 206-214.

USEFULNESS OF A NETWORK REFERENCE MONITOR.

Timothy C. Williams
Verdix Corporation
14130-A Sullyfield Circle
Chantilly VA, 22021

ABSTRACT

This paper will discuss the usefulness of a datagram level network reference monitor (*NRM*) that supports Mandatory Access Controls (**MAC**), Discretionary Access Control (**DAC**), Identification and Authentication (**I&A**), and Audit. It is based on the Verdix **VSLAN** which is a B2 level network component that enforces a security policy at the data link layer of the OSI protocol stack. Example architectures using non evaluated and a variety of evaluated hosts shall be shown and a discussion of the usefulness of the *NRM* in the architectures will be provided.

1. Introduction

The **VSLAN** is a B2 [1] network component that provides host systems with *network* mandatory access controls, discretionary access controls, identification and authentication of the hosts, and auditing based on the network security policy that has been composed by a network security officer. There are two components that work together to enforce the security policy prescribed by the network security officer. These components, when combined, create what is termed a *Network Trusted Computing Base* (NTCB).

1.1. Network Security Center

The Network Security Center (NSC) is the security officers interface to the **VSLAN** network component. From this machine (provided as part of the **VSLAN**), the network security officer can create the security policy for the network. The policy consists of both mandatory and discretionary access controls.

The network security policy is entered by the network security officer by selecting the allowable transmission and reception windows for the nodes of the network. The window of each node in the system is specified by a hierarchal level component (0 to 15) and a non-hierarchal compartment component (0 to 63). Also the network security officer can construct node access control lists to permit/deny the communication channels between nodes (independent of the level and compartment values). By using the combinations available, the network security officer can construct a multi-level network using single and multi-level host systems. These windows are assigned to a holder of a physical key (a *principal*). This person is authorized to initialize a particular node for a particular security window.

After the network security policy has been constructed and keys distributed, the nodes are initialized by their respective *principals* and the NSC downloads the security policy to the individual nodes. After the initialization process, the main purpose of the NSC is to collect audit data from the **VSLAN** nodes on the network.

[1] Currently under formal evaluation using the Trusted Network Interpretation (NCSC-TG-005) by the National Computer Security Center

1.2. Network Security Device

The Network Security Device (NSD) is the board level component of the **VSLAN** system. It provides the *NRM* function that this paper addresses. It is placed into the backplane of its host system and provides that host with its network access through the use of simple read/write operations. The host writes data to the NSD shared memory for transmission onto the network and reads data from the shared memory to access data that was sent to it from other hosts of the network. It is a multi-level device, meaning that a host can send multiple levels of data through the device and be assured that the label associated with the data shall be uniquely associated with that particular datagram until it is delivered to the destination host system.

It also provides assurances of origination of the datagram. Upon delivery of the datagram to the destination host system, the NSD writes the source node and authorized *principal* ID into the shared memory associated with the datagram. The host can use this data to determine authentication and identification of the user since the data (originating node and *principal* ID) was applied by the originating NSD which is a trusted device.

The NSD also provides cryptographically bound checksums for labels and data (the packet with the exception of the ethernet header is encrypted with the DES algorithm) to provide data integrity for the transmission of the data across the network.

1.3. VSLAN composition

A single **VSLAN** is composed of a single NSC and up to 128 NSDs. The NSC and NSDs communicate over an IEEE 802.3 compatible LAN (requires an 802.3 transceiver) which can be run over baseband, broadband, and fiber cables depending on site requirements. The current type of busses that the VSLAN supports are listed in table 1-1[2].

Table 1-1. **VSLAN** supported bus structures

Bus type	*Host systems*
PC AT	IBM PC AT, COMPAQ 286, most AT class machines
Q-Bus	DEC Microvax
Nu-Bus	Apple MAC II
3B2 bus	AT&T model 3B2 systems
VME bus	VME based workstations including SUN, APOLLO, INTERGRAPH, etc
MultiBus I	Gemini computer

2. Network Reference Monitor

The term *reference monitor* as defined by the the *Orange Book* [3] is:

(1) The reference validation mechanism must be tamper proof.

(2) The reference validation mechanism must **always** be invoked.

(3) The reference validation mechanism must be small enough to be subject to analysis and tests, the completeness of which can be assured.

[2] IBM, COMPAQ, DEC, Nu-bus, Apple, AT&T, SUN Microsystems, APOLLO, and INTERGRAPH are registered trademarks of International Business Machines, COMPAQ Computer Corporation, Digitial Equipment Corporation, Texas Instruments, Apple Computer, American Telephone and Telegraph, SUN Microsystems, APOLLO Computer, and Intergraph Computers respectively.

[3] The orange book (CSC-STD-001-83) is the guide by which the National Computer Security Center evaluated Computer systems.

The **VSLAN**, being a network component satisfies the definition in the following manner.

(1) Must be tamper proof - The hardware circuitry of the NSD prevents host systems from accessing any internal memory or devices (with special watchdog circuitry for enforcement even with hardware failures).

(2) Must always be invoked - The internals of the NSD are designed such that each packet must be processed by a task whose duty is to validate the labels and determine if the host is allowed to transmit or receive a particular datagram.

(3) Is small enough for inspection - By providing a datagram level of service to the host system, the NSD and NSC components are relatively *small* systems. Also, by being designed to meet the modularity requirements of B2 and higher level systems, inspection of the software and hardware portions of the devices is relatively simple.

2.1. A link layer network reference monitor

A link layer *NRM* provides a datagram service to its attached host system. The link layer (as viewed from the OSI protocol stack) rests between the physical layer (cable/bit movement) and the network layer (IP packet movement). It is at this layer which most *normal* ethernet boards perform their function. The host system writes data to the device to send to another host and reads data from the device to receive data that was transmitted from another host system. The difference between *normal* ethernet boards and the NSD is the type of processing that is performed on the datagram after the device gets the data. In *normal* ethernet devices, the device simply transmits the data onto the ethernet cable for reception by another node. Once the destination node receives the datagram that was addressed to it (or from any node if it happens to be in promiscuous mode) and simply passes the datagram to the host system. By allowing the host system to send to anyone it wants and/or receive from anyone it wishes opens a large door of opportunity for valid users of the system to access data to which they would otherwise not have access.

Since by definition, all network data must pass through the *NRM*, it can restrict access to the data that is transmitted over the network to only those which it was intended and delivery of datagrams only if they are within a prescribed security policy for the host system.

2.2. The importance of Discretionary Access Control processing

Discretionary access control processing at the link layer is based on a host having access to data that has been transmitted from another host system (independent of classification levels). If a host can receive/send data to all other hosts on the network, then its **DAC** list would include each host on the network. If the host can send/read data from just one host system, then that will be the only entry in the **DAC** list for that node.

Since the **DAC** processing will limit the availability of data based on a host to host basis, hosts can be assured that when a packet is placed onto the network, the only host system able to receive the data is the one for which it was originally sent. This creates a logical single wire between these two host systems.

2.3. The importance of Mandatory Access Control processing

Mandatory access control processing at the link layer is based on a window of security for each of the hosts on the network. Each host system has a window of security at which it can operate (a single level or multiple levels). The **MAC** processing allows host systems to communicate over a logical level **x** channel.

Since the **MAC** checking will prevent host systems from passing unauthorized levels of information, the network security officer can be assured that data that is labeled at a particular level can not be read by another host system that does not contain that particular level (or category) within its security window.

2.4. The importance of Identification and Authentication processing

Identification and Authentication in the operating system world consists of authenticating users (humans). In reference to a network component, the identification and authentication is done on a host basis. If the host system is identified and authenticated (with the use of a physical key inserted in an attached device), then it is given access to the network. If the process fails, then the host system is not given access to the network.

The identification and authentication of host systems is important (particularly in the ethernet medium) because it gives other hosts assurances that datagrams received from a particular address did in fact originate at that address.

2.5. The importance of Audit processing

Audit processing performed by the *NRM* can be critical for some environments. The logging and real time alarm processing capability of the **VSLAN** allows the security officer to be informed of attempted security violations at the time of the violation and not at some time later in the day (when some type of audit reduction processing can be performed). The audit processing of the **VSLAN** can trace each audit entry to the **VSLAN** host from which it originated, the particular level and category set of the packet that caused the attempted violation, and the *principal* of the node that caused the violation (among other information which can be used by trusted systems to identify the user process). Armed with this information, the network security officer can take appropriate action with respect to the individuals involved.

2.6. The importance of Data Integrity

Integrity of data that is moved from one host computer to another via a network is of great importance to most network system users. The **VSLAN** ensures data integrity by utilizing 3 cryptographically bound checksums and one 32 bit CRC polynomial checksum on each packet that is transmitted onto the LAN. The three crypto checksums are for security labeling, internal encryption header, and for the entire packet. These checksums are calculated and inserted into the packet before the packet is encrypted using the cipher block chaining mode of the DES algorithm. The 32 bit polynomial checksum is performed at transmission and reception by the 82586 LAN coprocessor component of the NSD. With these 4 checksums being performed on each packet, there is little chance that an un-detected transmission error will occur.

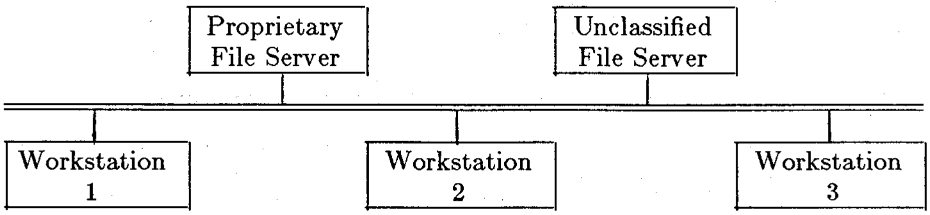
3. Using the network reference monitor

The use of the *NRM* in network systems can provide greater assurance of delivery and integrity of the data that is passed over the ethernet. The following paragraphs describe two architectures and what the *NRM* provides in each of the network configurations.

3.1. Simple network system

A common LAN configuration consists of a multiple single user workstations using another machine as a file/mail server. This LAN is usually between 5 and 50 nodes per group. A possible simple configuration of such an architecture would be 3 single user workstations (each running at a single level of classification) and 2 file servers (each running at a different level of classification). Figure 3-1 gives a diagram of such a network.

Figure 3-1. Simple network system



3.1.1. Constructing the simple network system

The composition of this simple LAN (possible automated office environment) would consist of 2 C2 rated file servers (to provide the restricted access to the files), multiple single user workstations (possibly running MS DOS and some of the more popular word processors and desktop publishing software), and the *NRM.* There will be 5 users of the network system and they are assigned the levels and workstations they can use by the network security officer as per table 3-1. The following paragraphs describe in more detail what each portion brings to the configuration.

Table 2-1. User to workstation mapping

Workstation	User	Level
1	Griff	Proprietary
2	Chris	Unclassified
3	Amy	Proprietary
1	Emily	Unclassified
2	Becky	Unclassified

3.1.1.1. Single level servers

Since the file server is rated at C2, it is assumed to be capable of providing controlled access to the files that reside on that device. It will provide a shared area of file storage for use by other users of the network system.

3.1.1.2. Single user workstations

The Single user workstation (possible unevaluated) is used a local computational engine. It provides the users with their preferred operating system (DOS, UNIX, VMS, etc.) and sends appropriate system calls to file servers for access to shared files.

3.1.1.3. Network Reference Monitor

The *NRM* provides the separate communication channels required to prevent data labeled proprietary from being read by a workstation/server with label unclassified. The *NRM's* will provide these communication channels between the servers and the appropriate workstations.

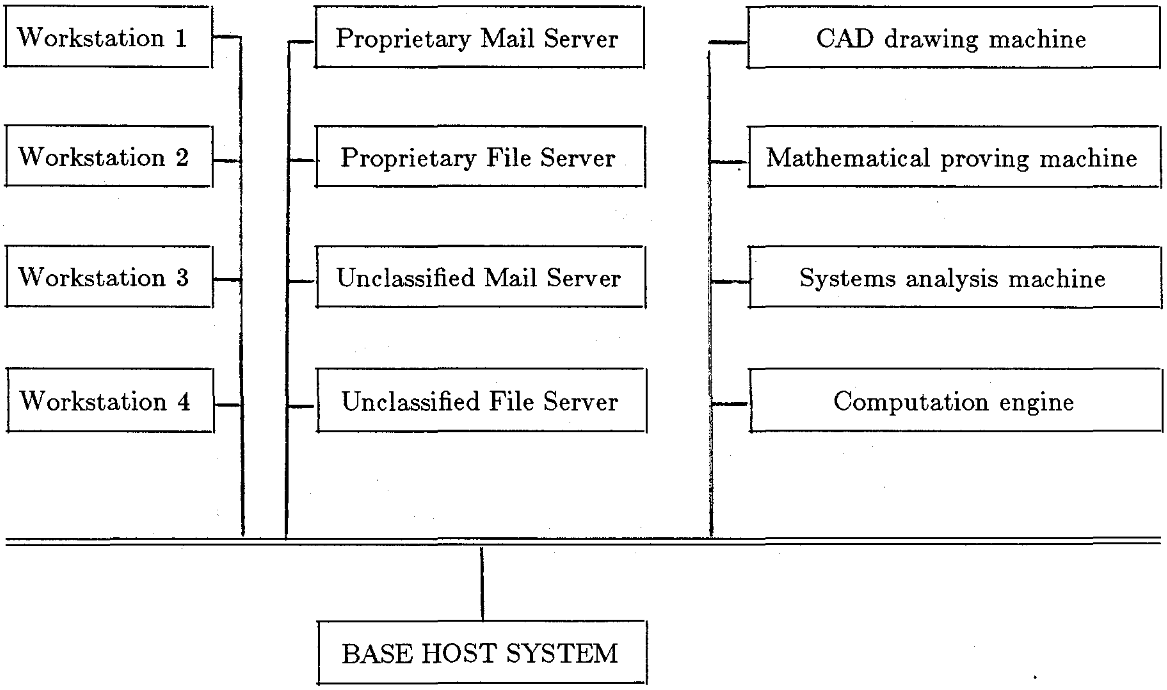
In the example, while Griff, Chris, and Amy (on workstations 1, 2, and 3 respectively) are communicating with their appropriate file server (Griff and Amy with the proprietary server and Chris with the unclassified server), the network security officer can be assured that data labeled proprietary will not be received by Chris or the unclassified file server. Also, if the DAC is setup to only allow workstation to file server communication, then the network security officer can be assured that Griff and Amy can not transfer data through the LAN from one workstation to another without checks performed by the file server.

To provide a cost effective solution for the utilization of resources, the ability to use a single non-evaluated machine as a workstation for 2 different levels of data (one at a time) can be very useful. In this example, when Griff initializes workstation 1 with his key, the security policy constructed for him by the security officer (communication with the proprietary server at level proprietary) will be downloaded to the node by the NSC. When Emily initializes workstation 1, her security policy will be downloaded to the node by the NSC (communication with the unclassified server at level unclassified). [4]

3.2. Complex network

A complex network (in this paper) is one that contains multiple host systems (each being used in a manner that best fits the type of system) that work together to fulfill the requirements of the installation in a secure and cost effective manner. The host systems are of a variety of vendors with a variety of ***trust*** assigned to them. The following discussion will describe the design and show how the *NRM* compliments the operating system security that is provided on the host systems. Figure 4-1 gives a pictorial view of the layout of the complex network system to be described. Figure 4-2 shows a list of users, their associated levels of clearance, and which workstation they are assigned.

Figure 4-1. Example of a complex network



[4] The use of the same machine for two different levels of security assumes that there are administrative policies and procedures in place for the removal of any storage media.

Table 2-1. User to workstation mapping

Workstation	User	Level	Compartment
1	Griff	Proprietary	Proof
2	Chris	Unclassified	F&A
3	Amy	Proprietary	Proof
3	Mike	Proprietary	SA
1	Emily	Unclassified	CAD
2	Becky	Unclassified	CAD

3.2.1. Constructing the complex network

In this configuration a MLS host systems' main purpose is to act as the user authenticator for the network system (with secondary functions of supplying a variety of file and computational services). Four other MLS host systems act as file/mail servers to store single level data but multiple compartment data. There are other project specific machines (single level and single compartment) that perform dedicated functions (CAD drawing, Mathematical proving, Systems analysis, and Finance and Administration). The final portion of this network are single user workstations used by the local network community.

The use of multiple levels of security on the machines elevates the complexity of the network design by an order of magnitude. Not only does the network design need to fulfill the basic functions requirements but it must address the security requirements as necessary.

In the design of this network system the abilities of the various portions of the network system must be considered. These components can be separated into three levels of operating system evaluations and the network reference monitor.

The three levels of operating systems that are used include non evaluated, C2 evaluated, and multi-level evaluated (B level and above). The multi-level systems are capable of authenticating users and providing multi-level and discretionary access to its internal objects. The C2 systems are capable of providing discretionary access to its objects. The non evaluated systems are not capable of providing any level of protection in a secure systems sense.

3.2.1.1. Base MLS host system

One of the most important portions of the design for the example system will be the selection of the B or higher rated[5] base host system that will be used to authenticate the network system users[6]. Once the user has identified himself to the machine, the other hosts of the network can be assured that the user is in fact who he claims to be. The other hosts can accept both the identification of the user and the level of information at which the base host system says that he can access. The machine should also have sufficient *horsepower* to give users acceptable response times. The base host system will be used mainly as an authentication device but should also be capable of multi-level file storage and mail facilities.

3.2.1.2. Single level servers

These devices allow the offloading of the file I/O intensive duties of the file server and mail server to designated machines. This separation of the file/mail servers allow other machines on the network to function in a higher capacity based on the offloading of this *slow* process. Since the

[5] The B level rating is required to be assured that the multiple levels of data that will be traversing the machine will be kept separated.

[6] This includes all single user workstations that may be on the network.

servers will be single level an argument can be made to use a C2 rated server. But this configuration allows the ability to put multiple compartments on the single level server. This requires the B level of trust to be assured that separation of the categories will be maintained. Since the main purpose of the MLS servers is to supplement the base MLS host's file and mail facilities, we can use a *smaller* base host system in conjunction with these servers to provide the users of the network system an acceptable response time.

3.2.1.3. Single level dedicated machines

These machines are used by the organization for specific purposes. The particular job for each host is related to the type of hardware/software that comprises the system. As an example, the Finance and Administration machine has the best COBOL compiler/operating environment for financial applications while the Mathematical proving machine has the tools/horsepower necessary for extensive number crunching applications.

3.2.1.4. Single user workstations

The last piece of the network is the unevaluated single user workstations. This portion of the network system allows the network designer to use cost effective devices to provide users with local processing and remote file accesses to the MLS hosts, single level servers, and project dedicated services that are available on the network. This portion of the network is vital because of the large numbers of these machines in use today and the cost/compatibility requirements of the network

3.2.1.5. Network reference monitor

The *NRM* is the glue that holds the network system together. They *NRM* provide logical communication channels over which the multiple levels of data can be transferred. It creates the equivalent of (16 levels * 64 categories * 2 [transmit and receive]) 2024 logically separate channels between 2 nodes of the system. In the complex system described, there would be a possible (16 levels * 64 compartments * 13 nodes * 2[transmit and receive]) 26624 logically separate possible communication channels.

The *NRM* provides the necessary assurances that hosts will not be capable of writing/reading data outside its prescribed security window and will not receive data (reguardless of level) from a machine that is not within its host access control list.

By using the *NRM*, the single user (single level) unevaluated workstations can be allowed to access network services without the possibility of compromise. The *NRM* will prevent data labeled outside the security window of the workstation to be received by that workstation. It will also prevent the workstation from accessing the individual hosts in the system without first authenticating himself with the base host system (he will not have any other host system in his host access control list).

The base host in turn can use the security level and associated ID of the sender of the datagrams as part of the authentication process (should it desire). The base host system can use the information received from the workstation to *lock in* the level of the user. This would not allow the user to change his level after he has been authenticated (the ability may have merit in some cases).

After the user authenticates himself, the base machine becomes logically a router for access to the other machines on the network. Since the base host system will have each host system in its access control list and the other host systems will only have the base system in their access control list, this will force the authentication of each user before network system services can be utilized. It will also assure that all data is labeled properly before delivery to the destination host system.

An example utilization of the network by a user, would be as follows:

Chris initializes his workstation and logs into the base host system (he is prevented for accessing any other host system through the use of the host access control list that is downloaded to his *NRM* at initialization time). He logs into the base host system (the base host system could use the label associated with the datagrams associated with Chris' transfers for determination of the classification level to be allowed) and the appropriate label is assigned to his process. Chris then accesses his mail through the use of the base host system services (which actually access the unclassified mail server) based on the classification and compartment values associated with Chris (Chris is prevented from accessing the proprietary server and unclassified data which does not contain his compartment set (F&A) by the *NRM* that resides in his workstation. At this point Chris can login to the F&A dedicated machine (because the F&A machine is unclassified with compartment F&A) to perform his normal work. It is important to note that Chris will be prevented from accessing the other dedicated machines because of his label containing the F&A compartment. The *NRM* will intercept these packets and not allow any such transfer to be accomplished.

3.2.1.6. Additions to the Complex Network

The *NRM* will help in the integration of other devices into the network system. If the network system requires access to the outside world, the *NRM* can be used to segregate the data and force certain levels of data to be sent to specific routers/gateways. This will ensure that data labeled proprietary (for the example system) will leave and come into the network via an encrypted router. While the unclassified data can use a router/gateway without encryption. In general, the *NRM* can be used to enforce the segregation of data within the network system and to devices that communicate to the outside world.

4. Wrap up

The *NRM* provides three basic building blocks for the construction of a secure networking environment (logical communication channels, data integrity, and a secure communication path between host systems).

The *NRM* provides logical secure channels by which host computers can communicate with a high level of assurance. These logical channels can be used by both multi-level and single level systems for secure communication.

The *NRM*'s use of cryptographically bound checksums throughout the datagram gives a high level of assurance of the label and data was transmitted correctly over the ethernet medium.

One of the most important abilities of the *NRM* is its ability to tell local systems the correct originating node and initializing user of each packet that it receives. This could play an important role in a constructing a secure network environment which contains single user workstations.

These facilities allow the network designer to provide the users of the network system a secure computing environment possibly without massive changes to the current computing resources. Thus the *NRM* is truly a ***building block*** upon which secure network systems can be constructed.

REFERENCES

(1) "Trusted Network Interpretation (NCSC-TG-005)", National Computer Security Center, 31 July 1987

(2) "Trusted Computer System Evaluation Criteria (DOD 5200.28-STD)", National Computer Security Center, December 1985

SAFEGUARDING PERSONAL PRIVACY AGAINST COMPUTER THREATS: A STRUCTURED PERSPECTIVE

Greg Young
18412 Gardens Way
Gaithersburg MD 20879

INTRODUCTION

Most of us have recently received 1990 census forms requiring personal information for government databases. There is no doubt that this information will enable more efficient government and business planning. For example, the allocation of federal funding to states and localities is based on census data. However, some have argued that such requests intrude on our individual privacy. While the census has been administered decennially since 1790, modern information system (IS) technology provides the means to access and use the 1990 Census in new ways that even proponents of efficiency and planning may not anticipate or advocate.

The census form prominently assures us that census personnel can be jailed for divulging individual records, and personal records will be so protected for 70 years. Obviously, the census administrators hope to mitigate popular anxiety concerning the collection and use of such personal data.

The timely example of the 1990 census illustrates issues of modern data collection processing, privacy concerns, and information safeguards that affects us all. The collection of personal information that uniquely identifies us as individuals presents us with an ethical, social policy, legal, and technical question: Are the steps taken to protect personal records sufficient?

The 1990 census is just one example of a database of personal records. Personal records easily can be used, stored, maintained, and/or distributed by government or business offices. These personal records may contain data describing a person's health, finances, purchases, criminal record, education, travel history, physical description, demographic description, or any other information that has been collected, stored and keyed to identify a unique individual. Technology now permits personal records in different systems to be aggregated, matched, retrieved, and shared in ways that were not authorized by the subject when the data was collected by the primary organization.

The rapidly increasing capabilities of modern computer technology for data storage and retrieval lends issues concerning personal records some immediacy. A twenty page file on each American can be maintained in 208 gigabytes of storage. This amount of storage is not uncommon for minicomputers or mainframes. One gigabyte of hard disk storage for a personal computer costs less than $5000 in current retail prices. In sum, current information system technology provides a feasible and often used means to gather personal information, often without the subject's knowledge. As of 1988, the federal government maintained over 3 billion files containing personal information on U.S. citizens (Craig 1988). These files are stored, manipulated, used, and shared with no substantive oversight concerning privacy issues.

Furthermore, the issue is not limited to government organizations. Data files of personal records in non-government organizations (e.g. banks' customer financial records, hospitals' patient health records, or marketing firms' demographic data and mailing lists) are subject to less regulation than government-owned files (Robinett 1989).

There is a potential danger that personal records may be applied manipulatively or otherwise against the subject individuals' interest (Southard IV, 1989).•For example, the Rand Corporation, working on a government contract, suggested that automatic teller machine records can be used to monitor the movements (date and time data) and transactions (debits and credits) of banking customers (Robinett 1989).

This paper examines the issue of information privacy and its status in the United States; specifically, the protection of identifiable individuals (data subjects) in the U.S. from the unauthorized use by third parties of hardware and software technologies to access data concerning those individuals. Popular literature has addressed computer "invasion of privacy", and public concern has led to legislative and regulatory protection in the U.S. and Europe. These prescriptions require organizations to make additional investments in the implementation and operation of information systems safeguards to protect against data security threats to individual privacy.

However, Europe has gone further than the U.S. in its legal requirements for the protection of information privacy (Farquhar 1985; Southard IV 1989). For example, the United Kingdom's Data Protection Act of 1985 includes in its requirements that:

1. private organizations inform a Data Protection Registrar concerning the sources, uses, and recipients of personal data;

2. personal data not be altered without the authorization of the individual;

3. personal data not be disclosed without the individual's consent; and,

4. violations of the Act are fined, with enforcement power in the office of the Data Registrar.

There is far less U.S. regulation and protection of personal data compared with the UK. Are the U.S. protections adequate?

While there has been extensive literature on information privacy, there is no taxonomy that relates the nature of the threat to privacy with the information system protective response. This paper:

(1) Reviews existing literature concerning data security as it relates to information privacy issues;

(2) Develops a taxonomy, or categorization, of these issues;

(3) Categorizes the information system response, function, and/or feature that protects against the information privacy threat; and,

(4) Relates legislative and regulatory prescriptions to the taxonomy developed in (2) in order to identify:

(a) information system responsiveness to social policy concerning information privacy; and,

(b) redundancy and/or gaps in social policy concerning information privacy.

This paper is intended to be a source document that integrates data security and privacy issues, the relevant social policy, and the appropriate information technology response.

THE AMERICAN INDIVIDUAL RIGHT TO PRIVACY

Privacy has a physical and an informational component (Southard IV 1989). In the example of the census discussed above, the census form is a physical instrument that has intruded in the individual's life for the purposes of collecting information. Informational privacy relates to the individual's desire to protect personal data from unauthorized use by others.

A person may wish to withhold some kinds of information from all others. The intimate details of one's personal life (e.g. sexual activity) is one example of such information. On the other hand, an individual may wish to protect some information only from those who potentially could use it against his/her interests. For example, a person may desire to give his/her medical information to a doctor, but not to an employer. Sensitivity of information, and the associated privacy concern, varies not only with its content but also its user.

As mentioned in the example of the census, the government needs personal information to fulfill its responsibilities. Similarly, private firms need personal information to efficiently compete in the marketplace. For example, a hospital needs to aggregate personal health records to plan its service provision; a video store needs to aggregate its rental transactions in order to stock films likely to be rented.

However, since 1891 the Supreme Court has recognized that the individual has a constitutional right to privacy (Southard IV 1989). In addition, tort law (the wrongful conduct of one person, including corporations, against another), although not uniform across the states, contains theories which may be applied to the protection of personal information against four types of violations (Scott 1987; Southard IV 1989):

1. intrusion, or unauthorized access;

2. appropriation, or interception, or "eavesdropping";

3. misuse of information, which includes use beyond reasonable expectation, libel or defamation, and the subject's right to know what information is included; and,

4. public disclosure of private information.

While the legal underpinnings of these four types of tort violations are not grounded in principles of information privacy, there is a body of tort cases that apply them to this area (Scott 1987; Southard IV 1989). The stream of these cases have developed the principle that information privacy can be violated only where there is a reasonable expectation that the information is private; and, the infringement of informational privacy should be balanced against opposing needs for operational need and organizational efficiency.

Modern data communications and the aggregation of personal information increases the frequency of events which may be characterized by one of the four violations discussed above. The next section of this paper examines the effect of IS technology on information privacy issues.

INFORMATION PRIVACY AND IS TECHNOLOGY

The rapid progress of modern information technology complicates the effort to balance rights and obligations concerning information privacy. The Supreme Court has held that the use of technology that is available to the general public by definition does not represent an illegal intrusion on privacy because, absent a minimum standard of privacy, a person should not expect to be protected against what is widespread (Southard IV 1989). By this reasoning, computer equipment, being generally available, can be used to collect, store, maintain and access data about individuals without representing any infringement on privacy. It follows, then, that as IS technology advances, the level of protected privacy will decline if safeguards are not mandated.

The case for protecting privacy from erosion by technology is based on several arguments. For one, a person may be inhibited in action if they feel that their actions are monitored or stored. For another, the benefits that a person may be entitled to often are subject to the information that is stored in their personal records. Finally, the burden of ensuring the integrity of personal information used by organizations is usually left to the individual affected (Southard IV 1989). The resources available to the individual typically are not sufficient to monitor the multitude of organizations which may be users of the information in question.

Modern computer technology also makes it difficult to monitor the accuracy and dissemination of personal records. Personal information, initially gathered by an organization with the permission of the individual (a primary organization), may be easily accessed or acquired by other (secondary) organizations that do not have permission of the data subject (OTA 1986). Personal information, stored on magnetic media, can easily be communicated; information in a computer system may be accessed online by remote organizations. Information provided with the subject's permission may be updated without the permission of the individual by attaching or scanning data obtained elsewhere. Over time, personal records in use may lose accuracy. In addition, the records may be used in ways not intended by the original authorization and may be used by secondary organizations (that is, those other than the original authorized collector, or primary organization).

The ethical or policy implications of IS intrusions on privacy rights is a key issue. For example, 75% of Americans think that government agencies that release personal records to other agencies are seriously invading privacy (OTA 1986) Moreover, as new applications come online, such as IRS aggregation of all of an individual's transactions (Scheibla 1985), it seems reasonable to expect the saliency of the issue to increase significantly.

The U.S. policy and legal framework regarding information privacy suffers from overlap, contradiction, and deficiencies (Hoerr 1988; OTA 1986; Regan 1988). In any case, the issue of information privacy is more than a legal one, it is also an ethical one, particularly for IS professionals (Auerbach 1985; Robinett 1989; Whieldon 1984). Whether the impetus to increase protection for information privacy comes from the law or IS professionalism, the integration of protective mechanisms into the information systems that use personal records will be the burden of the IS professionals who design, implement and operate these systems.

This discussion has observed that there is a developing principle of legal protection of personal information privacy, and that it is the ethical responsibility of the IS professional to implement protection of personal records. The next section of this paper examines the technical measures that are appropriate to protect privacy rights. It examines the personnel, technical, and economic approaches to protect personal records and information privacy.

AUTHORIZATION and MISUSE of PERSONAL RECORDS

The personal information "playing field" would be more level if individuals knew which organizations access their personal records. However, it is not reasonable to expect one person to monitor the multitude of organizations that access his/her personal records.

The executive branch of the federal government has implemented a notification order concerning requests for commercial information held by the government. This could be extended to protect individuals by implementing a reporting mechanism that notifies the individual when his/her personal record is accessed.

Notification gives the individual the opportunity to check the accuracy of the information as well as knowledge of which organization is using their record. Along with the notification, then, goes the individual's right to access and correct their own personal record.

Such individual notification would need to be constrained by public safety considerations (e.g. criminal investigations). Still, the burden on reporting organizations may be significant. Therefore, the notification requirement could be limited to secondary organizations (organizations that are not the original authorized data collector) that access specific personal and unique identifiers that directly intrude on an individual's privacy (e.g. name, address, or social security number).

It is noted here that it is current practice of the federal government to notify the data subject of his/her Privacy Act rights whenever collecting personal information. This practice could be extended to the private sector. The following suggestions to protect personal information privacy extend both to the private and public sectors:

Privacy Safeguard 1: Implement a reporting mechanism that notifies the individual when his or her personal record is accessed by other than the primary organization.

Privacy Safeguard 2: Provide an opportunity for the data subject person to examine, update, or delete information contained in his/her personal records before the record is used in organizational processes. The best opportunity is at the time of the notification process discussed above.

Privacy Safeguard 3: Notify the data subject of their legal rights at time of initial data collection.

The notification and data subject update procedure will place a burden on the processing organization. The organization will have to devote resources and establish procedures to involve the person who is the data subject in the life cycle of the personal record. For example, this person may initiate or approve maintenance activities such as modification or deletion of erroneous personal data. The individual may also be involved in the authorization of personal data dissemination. The processing organization will be responsible for compliance with these constraints.

Privacy Safeguard 4: Legislate a requirement that organizations appoint a data protection officer to preserve the privacy of personal records.

The additional burden on the organization may be required if information privacy legislation in the U.S. develops along the European model (Janner 1984; Korn 1985; Regan 1988; Riley 1985). For example, the English, to enforce compliance with their Data Protection Act of 1984, have mandated that data-using organizations designate a data protection officer who has "hands on" all applications. In addition, there is a governmental data registrar, or "czar" with the power of enforcement. However, in the U.S. it may be that the popular sentiment against "layers of bureaucracy" will make it more expedient for the function of "czar" to be left with the courts, at least initially.

IS SECURITY of PERSONAL RECORDS

The previous discussion has focussed on the use of personal records by secondary organizations. The assumption has been that such use is authorized by the primary organization, although not by the subject individual, in keeping with the current policy that less than rigorously protects the individual. However, there also may be illicit dissemination, access or interception. In a policy environment that is sensitive to the protection of information privacy, then, security measures that are generally applicable to secure "trusted" IS also are appropriate to protect personal records. These measures include:

1. administrative security (password protection, security training, management support and resource allocation for security);

2. physical security (locked and secure facilities and data); and,

3. system security (online audit systems, secure network interface, protected containment of personal records).

It has been noted that in the current environment it is not feasible for the individual to monitor all activity concerning his/her personal records. Similarly, the feasibility of shifting the burden of securing an enhanced concept of information privacy to the processing organization needs to be considered. The security measures listed above should be considered from this perspective.

Privacy Safeguard 5: Implement economically justifiable administrative security to personal records.

Privacy Safeguard 6: Implement economically justifiable physical security to personal records.

Privacy Safeguard 7: Implement economically justifiable system security to personal records.

The organizations that access personal records are more likely to be called on to defend their activity in a social and legal environment that is more sensitive to the protection of information privacy. Therefore, the internal cost for one organization to implement the protection of personal records should also reflect the potential liability, legal costs, and value of lost "good will" if the security of personal records is violated. These costs could be considered in the allocation of resources to the security activities outlined above. This argument for security protection of personal records has an economic basis.

This discussion has examined legal and technical issues associated with information privacy. The next section integrates these and uses this framework to add insight to the examination of information privacy.

A STRUCTURED PERSPECTIVE OF INFORMATION PRIVACY

This section categorizes IS activities and integrates the categorization with critical areas of information privacy. This structured framework identifies those areas in which currently there is a gap in coverage. These gaps strongly suggest the need for additional consideration from IS professionals and social policy-makers who are concerned with protecting information privacy.

Table 1: Integration of Information Privacy and IS compares the legal rights of data subjects concerning information privacy (the columns in Table 1) with the IS activities that process personal records (the rows). Each column is subdivided into current status in the U.S. (C) and safeguards proposed in this paper (P). The cell values identify the number of the appropriate Privacy Safeguard as identified in this paper and summarized on the next page.

LEGAL RIGHTS CONCERNING PRIVACY

IS ACTIVITY	Privacy		Access		Accuracy	
	C	P	C	P	C	P
Collect		4	3*	1,3,4	3*	2,4
Maintain		4,5-7		4		4
Use		4		1,4		4
Disseminate	3 +	4		1,4		2,4

Table 1: Integration of Information Privacy and IS

3*: The Privacy Act targets government IS only.
3 + : The Privacy Act requires notification before dissemination, but there are many loopholes, and no oversight; government coverage only.

Table 1 indicates that information privacy rights are currently not rigorously protected by IS activities. For example, the privacy of data is not protected during the collection, maintenance and use activities. The access and accuracy of personal records is not protected during the maintenance, use, and dissemination activities.

The Table associates the recommended safeguards proposed in this paper with the aspect of information privacy and the IS activity it covers (column P in Table 1). A comparison of the current protections with those proposed shows that more complete coverage is recommended. Moreover, the comparison shows that there are gaps, or areas not addressed with current policy and practice, in the security of personal records. This paper proposes that these gaps will need to be addressed in a society that is more concerned with information privacy.

DISCUSSION

The U.S. will not be able to soon address the issue of information privacy with the rigor of European social policy. This is because there are numerous initiatives, laws, and policies in this country, but they overlap and contradict one another (OTA 1986). While the U.S. Congress has proposed numerous study commissions and regulatory offices to address information privacy issues, there are still opponents who are concerned with centralization of information control. The OTA (1986) has recommended continued study on the issues of information privacy.

However, there has been legislation introduced in the House of Representatives which goes further. H.R. 1721 in the 99th Congress would establish a Data Protection Board. The bill gives the Board authority over personal records in the Federal government, to develop guidelines, investigate compliance, intervene in agency proceedings, and issue subpoenas. Directors of the Board would be appointed by the President with the consent of the Senate.

While this bill has not become law, it does embody a first step in the direction of information privacy such as is found in the UK Data Protection Act. A Data Protection Board that extends its authority to the private sector would more closely approximate the protection found in the UK law.

SUMMARY

This paper has identified specific areas concerning information privacy that can be addressed by IS professionals. These include:

1. Notification to individual when an organization accesses personal record with name, address, or social security number identifier.

2. Provide an opportunity for the data subject to examine, update, or delete information contained in his/her personal records before the record is used in organizational processes.

3. Notify the data subject of their legal rights at time of initial data collection.

4. Legislate a requirement that organizations appoint a data protection officer to preserve the privacy of personal records.

5. Implement economically justifiable administrative security for personal records.

6. Implement economically justifiable physical security for personal records.

7. Implement economically justifiable system security for personal records.

It is encouraging to note that several companies have undertaken their own initiatives in this area (Riley 1985). For example, IBM has disseminated guidelines for handling personal data which address accuracy of information and adequacy of security. However, the adequacy of guidelines to secure personal records in the absence of external pressure (such as legal requirements) to compel implementation is an open question. Another firm, Air Products & Chemicals, has found that none of its departments has 100% compliance with security standards to protect critical data stored on personal computers (Scheier 1990).

In addition to initiatives from government and individual firms in the private sector, there are professional associations, such as the American Federation of Information Processing Societies, which study issues of national information policy, including privacy. The problem, however, is implementation, not awareness.

APPENDIX: GLOSSARY OF INFORMATION PRIVACY

Information privacy concerns are not new (reference the popular literature, e.g. Orwell's 1984). A cursory online scan of business journal's for abstracts containing the combinations of "computer privacy" and "personal" shows over 250 citations since 1984. Still, for reasons not clear to this writer, information privacy has been of more concern, and is more protected, in Europe than in the U.S. (Collier 1985, Farquhar 1985; Janner 1984; Linowes 1985; Pounder 1985; U.S. Congress 1986). It is appropriate then, to briefly review the terminology concerning information privacy.

Authorization: The explicit permission given by data subjects to each user of data which identifies that individual data subject.

Data Subject: A living individual identifiable by computer-based data, see subject data, personal record.

Information Privacy: the protection of individuals (data subjects) from the unauthorized access of data which uniquely identifies those individuals.

Personal Record: Data concerning an individual that uniquely identifies that person, see subject data.

Primary Organization: an organization that collects personal data from the subject individual with permission, and uses the data to initiate a personal record.

Primary System: An information system that collects information from a person and creates subject data, or a personal record that uniquely identifies that individual.

Secondary Organization: an organization that accesses or uses personal records that were disseminated from other than the subject individual.

Secondary System: A system that uses or accesses personal records but is not the original, or primary, collecting system.

Subject Data: A personal record that uniquely identifies an individual, typically by name or Social Security Number.

BIBLIOGRAPHY (This bibliography uses the format required by the Academy of Management Review.)

1. Auerbach, I.L. 1985. Professional responsibility for information privacy. Journal of Information Systems Management, 2(1), 77-81.

2. Betts, M. 1986. Safeguarding privacy: MIS confronts a sensitive challenge. Computerworld, 20(27), 53-54, 56-57, 59, 60.

3. Bigelow, R.P. 1986. Computers and privacy -- An American perspective. Information Age (UK), 8(3), 134-140.

4. Biskup, J. & Bruggemann, H.H. 1988. The personal model of data: Towards a privacy-oriented information system. Computers and Security (UK), 7, 575-597.

5. Collier, G. March, 1985. Data privacy: New law puts onus on firms. Chief Executive (UK), 75-76.

6. Craig, B.T. 1988. Bits, bytes and freedom. Executive Speeches, 2, 20-23.

7. Diamond, S. 1987. Talking security: Foiling the data swindlers. Wall Street Computer Review, 4(12), 44-54.

8. Farquhar, W. 1985. Access control for computer systems. Information Age (UK), 7(1), 25-29.

9. Field, A.R.; Neff, R.; Seghers, F.; & Deveny, K. Feb. 9, 1987. 'Big Brother Inc.' may be closer than you thought. Business Week (Industrial/Technology Edition), 84-86.

10. Graham, J.P. 1987. Privacy, computers and the commercial dissemination of personal information. Texas Law Review, 65, 1395-1439.

11. Harris, D. 1987. A matter of privacy: Managing personal data in company computers. Personnel, 64, 34-43.

12. Honan, P. 1987. Data security. Personal Computing, 11(1), 101-107.

13. Hoerr, J. March 28, 1988. Privacy. Business Week (Industrial/Technology Edition), 61-68.

14. Janner, G. 1984. Data protection: Ignorance will be no excuse. Accountancy (UK), 95(1094), 87-88.

15. Katz, J.E. 1988. Public policy origins of telecommunications privacy and the emerging issues. Information Age (UK), 10(3), 169-176.

16. Kirby, M.D. 1986. Privacy protection in Australia: An update. Information Age (UK), 8(4), 200-207.

17. Korn, A. 1985. Time for action on data protection. Personnel Management (UK), 17(12), 40-43.

18. Leahy, P.J. 1986. Privacy and progress. Computers & Security (Netherlands), 5, 347-349.

19. Linowes, D.F. 1985. Privacy protection in the United States in 1984 - Is it adequate?. Vital Speeches, 51(6), 168-171.

20. Marion, L. 1986. Information technology: The rocky road to globalization. Institutional Investor, 20(10), 279-284.

21. Moad, J. 1987. As AIDS spreads, state PC systems are reaching limits. Datamation, 33(16), 43-56.

22. Oley, L. 1987. Security: Trusted applications. Systems International (UK), 15, 60-61.

23. Pfleeger, C.P. 1989. Security in computing. NJ: Prentice Hall.

24. Pounder, C. March, 1985. The data protection problem. Management Today (UK), 39-46.

25. Regan, P. 1988. From paper dossiers to electronic dossiers: Gaps in the Privacy Act of 1974. Office: Technology & People (Netherlands), 3, 279-296.

26. Regan, P.M. 1986. Privacy, government information, and technology. Public Administration Review, 46, 629-634.

27. Riley, T. 1985. Setting standards -- Data privacy. Computer Data (Canada), 10(12), 12-13.

28. Robinett, J. 1989. Private lives are in MIS's hands. InformationWEEK, (212), 52.

29. Scheibla, S.H. 1985. Open secret? More and more people have access to your tax return. Barron's, 65(25), 13

30. Scheier, R. 1990. Firm protects critical PC data with mainframe-style policies. PC Week, 7(15), 1,6.

31. Scott, M.D. 1987. Computer Law. New York: John Wiley & Sons:Wiley Law Pubs. 7-34 to 7-38.

32. Southard IV, C.D. 1989. Individual privacy and governmentalefficiency: Technology's effect on the government's ability togather, store, and distribute information. Computer/Law Journal,IX, 359-374.

33. Stoll, C. 1989. The Cuckoo's Egg. New York: Doubleday.

34. Taraz, A. 1985. Data protection: How the Act will affect you.Accountancy (UK), 96(1103), 122-124.

35. U.S. Congress. 1986. Office of Technology Assessment, Federal government information technology: Electronic record systems and individual privacy, OTA-CIT-296 (Washington, D.C.: U.S. Government Printing Office).

36.Whieldon, D. 1984. Ethics: MIS/DP's new challenge. ComputerDecisions, 16(13), 92-110.

Legal Issues in Security & Control of Information Systems

Noah Stern
803 Karen Court
Apt. 104
Laurel, MD 20707

The evolution of computers and information systems is changing our lives on an almost daily basis. Everything from a person's standardized test scores to his/her financial history is stored in an information system somewhere. The range and personal nature of this information tempts unscrupulous people to abuse it for personal gain. While computers and information systems have an enormous capacity to help society, they can also be used in ways that hurt society. Because of this potential, adequate "computer security" safeguards are an essential component of computers and information systems. This includes using the legal system to prosecute computer malefactors.

Do special laws need to be enacted for the purpose of protecting computers and information systems or are the current laws adequate? What issues do computer security laws need to address? The purpose of this paper is to outline the legal issues involved in security and control of information systems.

Specifically, the following topics will be addressed:

1. Brief history of computer crime and what constitutes a computer crime.

2. Laws already in place to protect computers and information systems and their effectiveness.

3. issues related to businesses protecting information systems and data.

4. Legal liability when confidential data is misused.

HISTORY AND DEFINITION

The first reported computer crime occurred in 1958. The first federally prosecuted computer crime involved alteration of bank records in Minneapolis in 1966.[1] In many of the early cases, the computer crime went undetected for a long time. For example, from 1970-73, a bank teller embezzled $1.5 million from his bank employer. The teller transferred funds from inactive accounts into his own, manipulated adjustments and failed to enter transactions periodically (the last activity is the most difficult to detect). The debits and credits balanced, so the system auditors did not detect any wrongdoing. He was caught when the police raided his bookie, whose records showed that he bet $30,000 a day![2] In another case, Equity Funding Corporation of America embezzled $2 billion from 1964 to 1973. Part of the scheme involved using a computer to issue bogus insurance policies. Finally, the Illinois Insurance Commission detected the scheme during a surprise audit.[3]

Initially, many of the most publicized computer crimes involved embezzlement, but computer crime encompasses more than just white-collar criminal activities. Computers have been used to help commit theft, and espionage. For example, the Walker case involved selling classified Navy secrets gleaned in part from a computer.

Paramount to understanding the issues surrounding computer crime are definitions of pertinent terms. Computer Abuse is defined as "the broad range of international acts involving a computer where one or more perpetrators made or could have made gain, and or more victims suffered or could have suffered a loss."[4] Computer Crime is defined as "illegal computer abuse. It implies direct knowledge and involvement of computers in committing a crime."[5] Computer Related Crime is defined as "any illegal act for which knowledge of computer technology is essential for successful prosecution".[6] Retired Senator, Abraham Ribicoff (D. Conn.) identified four primary categories of computer-related crime [7]:

1. Introduction of fraudulent records or data into a computer system.

2. Unauthorized use of computer-related facilities.

3. Alteration or destruction of information or files.

4. Stealing money, financial instruments, property services or data.

CURRENT COMPUTER LAWS

Several already existing non-computer-specific laws are used to prosecute computer criminals. As previously stated, computers are used to commit many kinds of crimes. Consequently, state and federal laws enacted to protect against those crimes apply to criminals who break these laws while committing computer crimes. The following is a brief overview of some non-computer-specific laws and how they are used to prosecute computer criminals.

State Non-Computer-Specific Laws

Criminal Mischief. Definition - willful destruction of the property of another. This offense only applies to hardware and not software. Although traditional means can be used to detect and appraise hardware damage, it is much more difficult to detect and assess the damage caused by, for example, a logic bomb. Several state jurisdictions have statutes for criminal mischief.[8]

Burglary. Definition - unlawful breaking and entering a building with intent to commit a crime; sometimes, the conduct must occur at night. Almost all jurisdictions have statutes regarding burglary. They can effectively be used to prosecute individuals who gain unauthorized access to a computer facility. However, accessing a computer via a remote terminal is not burglary. Burglary cannot be used to prosecute crimes involving remote access, because ordinarily no physical trespass and entry of a building has occurred.[9]

Larceny. Definition - felonious taking of the personal property of another person without his consent and with the intention of permanently depriving him of it. This charge is easily applied to the theft of computers, tape drives, and other hardware, but is more difficult to apply to software. "Pirating" a program does not deprive the original owner of its use, but it may be held the felonious taking of the personal property of another person without his consent. New York's larceny statute recognizes property as "any article, substance or thing of value" [N.Y. Penal Law, sec. 155.00(1)]. Texas state courts have also viewed software as property [Susan H. Nycum, "Legal Sanctions to Computer Abuse," Assets Protection, vol 2, No. 3, 1977, p.33.].[10]

Theft/Misappropriation of Trade Secrets. Definition - unlawful taking of "secret scientific material" .[11] This category will be addressed in a later section of the paper.

Embezzlement. Definition - unlawful appropriation of another's property by one who has lawful possession of the property. this charge has been frequently used to prosecute people like the gambling bank teller mentioned above. Perhaps, more computer criminals have gone to jail as a result of embezzlement charges convictions than for any other charge.[12]

Receipt of Stolen Property. Definition - this requires that the receiver know or reasonably suspect that the property is stolen and intend to deprive the lawful owner of its use. Software provides the same problem here as it does with larceny.[13]

Interference with Use. Definition - unauthorized interference with, tampering with, or use of another's property that causes a loss to the lawful owner. This statute is used effectively to prosecute against obliteration of computer software or data. It would also appear to cover infecting another person's data with a virus or worm.[14]

Forgery. Definition - falsifying or materially altering, in order to defraud, any writing where the writing, if genuine, implies a legal liability. Any hacker who breaks into a system and shifts money to his account could be prosecuted for forgery.[15]

In addition to the above non-computer-specific laws, many states now have statutes that specifically address computer crime. California even protects programmable pocket calculators.[16] When a case is prosecuted by a state, it is not uncommon for the state to use a combination of computer-specific laws and conventional laws.[17]

Federal Non-Computer-Specific Laws

Several of the crimes that state statutes cover are addressed by federal law as well. However, these laws tend to be narrower in scope, in that they usually require the involvement of federal property and/or the crossing of state lines.[18]

Conspiracy. Definition - the agreement between two or more individuals to commit an unlawful act. (18 USC 371) This is a very broad statute, but the U.S. Supreme Court has held that it is constitutional because the absence of such a law presents a danger to the country. Because banks and other financial institutions are members of the FDIC or the FSLIC, crimes against these banks can be prosecuted using this statute. However, a single person acting alone cannot be federally prosecuted for conspiracy.[19]

Forgery. Definition - passing, publishing or selling "any falsely made, forged or counterfeited security of the United States" with intent to defraud (18 USC 472 & 473). This statute could be used to prosecute software pirates. An attempt to use this statute against an Electronic funds Transfer System attack would probably not succeed because no forgery per se occurs.[20]

Embezzlement and Theft. Definition - embezzling, stealing, knowing conversion or the receipt, concealment, or retaining of "any record, voucher, money or thing of value" of the Federal Government with intent to further personal use or gain (18 USC 641). this is a broad statute that could cover many types computer crimes.[21]

Mail Fraud and Wire Fraud. Definition - the use of the mails and wire services in the course of committing fraud (18 USC 1341 & 1342). The mail fraud statute only applies to the U.S. Postal Service. It makes no mention of private mail services like Federal Express, for example. The Wire Fraud Statute applies to telephone, telegraph, radio or television communications in interstate commerce with regard to any fraudulent scheme.[22] In 1976 the Wire Fraud Statute was used to prosecute Bertram Seidlitz, a former employee of a computer firm located in Maryland. Seidlitz, while still employed with the firm, obtained some secret computer access codes. Seidlitz subsequently resigned and accepted employment from another company in Virginia. From his new job's office, Seidlitz dialed via modem into his previous employer's computer for the purpose of stealing software. He continued to access his former employer's computer illegally for several months before he was caught. Because Seidlitz dialed in from his Virginia office (across state lines) instead of his Maryland home, the prosecutor successfully applied the Wire Fraud Statute.[23]

Interception of Wire or Oral Communications. Definition - the unlawful willful interception or attempt at interception of any wire or oral communication (18 USC 2511). This law was intended to discourage wiretapping, among other things. The authors had in mind human conversation over wires, so there is some question as to whether intercepting machine language could be prosecuted under this statute. Another part of this same statute prohibits the use of "any electronic, mechanical or other device" to assist in this crime. Since it requires a computer to decode bits sent over the wires, it may nevertheless be appropriate to apply this statute.[24]

Federal Computer-Specific Laws

While several of the federal non-computer-specific laws are used successfully to prosecute crimes involving computers, there are important issues that they do not address. As a result, Congress enacted a series of laws that focused more specifically on computer security issues. These laws attempt to close some of the "holes" in existing laws and address problems that occur only with computers.

The Privacy Act of 1974. This law was passed to protect private citizens from having their personal data in federal data banks used for unintended purposes. For example, this law prevents a bank from calling up the IRS for information from 1040 forms when someone applies for a bank loan. It is worthy to note that heads of federal agencies can exempt their agencies from this law if they think it is necessary.[25] This is not the only statute that applies to privacy and security aspects of computer crime. Several states have such statutes also. This issue will be readdressed in the last section of this paper.

The 1986 Federal Computer Fraud and Abuse Act. This statute prohibits the unauthorized access or use of computer systems in three areas. First, the Act makes it a felony to use or access a computer, without authorization, to obtain classified U.S. military or foreign policy data with intent to harm the interests of the U.S. or to benefit a foreign nation. Second, the act makes it a "misdemeanor to obtain financial or credit information protected by federal financial privacy laws". Third, the Act makes it a misdemeanor to access a federal government computer, without authorization, and or modify, delete, or disclose any data the computer contains, or interrupt service.[26]

The Computer Security Act of 1987. This act empowered the National Bureau of Standards to develop computer security guidelines in coordination with other agencies and offices, including but not limited to NSA, DOD, GAO, OTA, and the OMB. The Computer System Security and Privacy Advisory Board was established to oversee the process. the Act mandated that once the guidelines have been established, all federal and federal related computer systems must comply with them. However, the guidelines can be disregarded by the certain agency heads and the President. Additionally, each federal agency is required to provide computer security training to all personnel who are involved with the use or operation of its computers. Finally, the Act requires all federal agencies to identify all of their computers that contain sensitive information within six months of the Act's enactment. In conjunction with this activity, each agency has one year to establish a computer security plan for those computer systems identified.[27]

The Computer Matching Privacy act of 1988. The purpose of this Act is to "regulate the use of computer matching conducted by Federal Agencies or using Federal records protected by the Privacy Act of 1974".[28] "Computer matching" means searching one computer's online/offline storage for information based on information stored in another computer. More specifically, computer matching is the cross-indexing of data based on field values (a Social Security number for instance). The Act specifies that no agency may have access to data from a non-federal agency without a written agreement. The agreement must specify the purpose of the program, be justified in terms of results and estimated savings, and contain a description of the records to be matched. Additionally, individuals under the jurisdiction of the agency must be given notice that they can be subjected to computer matches. The agreement must also specify procedures for insuring the security of the records, prohibitions on duplication and redisclosure of the records, procedures governing how the records will be used, information on the accuracy of the records being matched, and finally, that the Comptroller General may have access to records necessary to enforce compliance of the agreement.[29]

The Computer Virus Eradication Act of 1989. This bill (H.R.55) was recently introduced to amend 18 USC 1030, and is currently in the House Subcommittee on Criminal Justice. The bill would add language that specifically mentions computer viruses. It would make criminal those actions with viruses that may cause loss, expense, or risk to health or welfare. Additionally, it gives victims civil remedies.[30]

A major problem in prosecuting computer crimes is that law enforcement personnel are unfamiliar with the current technology. The FBI is attempting to remedy this situation with its own personnel. Federal agents are attending seminars and classes designed to help them recognize what computer crimes are and how they are committed. Unfortunately, local law enforcement personnel commonly have rudimentary computer expertise, at best.[31] Also, many computer crimes do not fit neatly into the existing laws. Computer technology is constantly evolving, and current laws may be inadequate for future technologies. To address this problem, the 1986, 1987, and 1988 federal Acts use broad definitions with respect to computer crime. This is resulting in a case-by-case examination of these crimes to establish precedents. There are two apparent weaknesses, however, with regard to these statutes. The penalties are not as severe as they should be and they do not appropriately distinguish between degrees of malevolence with regard to the crimes.[32]

LEGAL PROTECTION OF BUSINESS INFORMATION SYSTEMS AND TRADE SECRETS

Businesses have several major problems concerning the protection of information systems and data. If they have contracts with the federal government and access to sensitive data, they are now required to protect that data. If they have access to federally protected financial data, they are required to protect it also. Additionally, companies can invest several years and many dollars training an employee, only to have the employee leave to work for a competitor. If the employee is a programmer, more than just data and trade secrets may be lost. The question therefore rises whether a company can protect its investment in time and money in training programmers and other employees.

The first and most important thing a company can do is to protect itself by contracts, which can cover ground not covered by statutes. The company should have all new employees sign employment agreements containing noncompetition clauses. The noncompetition clause must be for only a reasonable period of time and must cover only a reasonable geographic area[33]. An agreement that is too restrictive could be voided in part or in whole by the courts. In Solari Industries vs. Joseph Malady, a noncompete clause was reduced in scope because the geographic area covered was too broad. Joseph Malady signed an agreement with Solari Industries that restricted him from selling a competing product for one year. The agreement did not specify a location, and Solari took the position was that the agreement covered the entire United States. The courts disagreed and allowed an injunction to be applied only to the New York City territory.[34]

It has been my experience in the programming industry that one year is the normal time frame for a noncompetition agreement. That is, the employee will not compete with the former company for one year after he/she leaves the company. These agreements should also be applied to consultants and subcontractors that work for the company.[35]

Also, a company may consider including a "repayment clause" in the employment contract. The repayment clause is a controversial way to cover the cost of training. Nevertheless, some companies routinely include repayment clauses in their employment contracts (for example, Electronic Data

Systems Corp.). A repayment clause should be based on a reasonable estimate of the cost of training. A competent lawyer should be consulted when drafting such an agreement.

It is considered "good form" to inform interviewees that signing an employment agreement is part of what is expected. An employee is likely to be resentful if he/she shows up for the first day of work, and with no notice, is presented with a form and told further employment is contingent upon signing the form. this practice is likely to make a bad first impression with the employee, particularly if the employee has just moved to another part of the country. Also, current employees should not be asked to sign one of these contracts unless they are given additional benefits or other consideration. Some courts have held that without additional consideration, an employee cannot be forced to sign away a right.[36]

When drafting an employment contract, a company should strive to avoid making "unreasonable" demands on prospective employees. The courts generally do not enforce labor contracts that place undue burdens on the employee.[37] EDS is currently embroiled in such a lawsuit with a former University of Maryland Computer Science graduate. The student interviewed with EDS for a job. He went to a follow-up interview and was told that he would not be hired unless he signed an employment agreement. The employment agreement essentially contained the following: EDS will teach Cobol, Job Control Language (JCL), and the EDS way to write Cobol code. If the student leaves EDS within one year, he owes EDS $5000, if he leaves within two years, he owes EDS $3000. If the student leaves EDS between the second and the third year, he owes EDS $1500.[38]

The student signed the agreement and went to work for EDS. According to the employee, EDS proceeded to demand that he work 50+ hours per week. After a year and a half, the employee resigned and refused to pay EDS. EDS sued and the case went to court. The defendant alleges that he knew Cobol and JCL before employment with EDS and that the EDS programming way is nothing more than a routine flowchart algorithm and hardly provides any grounds for repayment of $3000. Furthermore, the defendant alleges that EDS tried to use the contract to impose a form of involuntary servitude on him. EDS alleges breach of contract. Each side has pledged to pursue this action vigorously, and the case is still pending.[39] EDS has litigated several of these cases in the past.[40] The fact that EDS has sued several times with respect to this indicates employee unwillingness to abide by this scheme. They appear to consider it overreaching. Perhaps, EDS needs to reconsider the approach of its employment contracts.

Another good practice to identify security weaknesses is to conduct exit interviews when employees leave the company. There is certain to be some amount of "sour grapes", but some hitherto unknown weaknesses may be identified.[41] As with other security precautions, a good lawyer will advise a company to implement preventive measures first and litigate second.

Maintaining the integrity of trade secrets is an important area of computer security with respect to businesses. What is a trade secret? A trade secret is hard to define, but the following description may be helpful: "any formula, pattern, device, or compilation of information which is used in one's business, and which gives him an opportunity to obtain an advantage over competitors who do not know or use it".[42] The issue of trade secrets arises because the courts usually interpret rights to programs under copyrights and/or trade secret law. The U.S. Patent Office has until recently been reluctant to grant patents on software, although in 1981 it granted one to a software brokerage house.[43] Patents are effective only for seventeen years and the patent owner must vigorously prosecute patent infringers to suppress patent infringement. This prosecution can be time consuming and expensive, often hundreds of thousands of dollars. Additionally, the Patent Office requires a thorough description before it will issue a patent. This can enable the competition to understand how it works.[44] The entire process may take three or more years and cost five to thirty thousand dollars.[45] As a result, most companies resort to using copyright law and trade secrets, instead. That is, they do not patent a new idea or invention (software), but rather try to keep it secret from the competition, or else copyright the program.[46] Software was considered a trade secret in the

Seidlitz case.[47] Information can also be considered a trade secret by the courts. Often a company keeps information about its trade secrets on its computer.

This issue is important because large sums of money are often at stake in these cases. Sometimes the survival of the company is in jeopardy if a trade secret/copyright/program is lost. The question arises over and over again, "Who owns what is in your head?"[48] More important, how do you protect it?

Stanley H. Lieberstein, a New York patent attorney and author of Who Owns What is in Your Head?, recommends that shredding all paper trash and cleaning off desks at the day's end are two simple measures to protect confidential information. He relates that numerous trade secrets have been lost through study of the trash and from people seeing things they should not have, on other people's desks.[49]

Trade secret law is not new. A recorded case occurred in 1414, the Dyer's case.[50] Consequently, several trade secret legal concepts have been in place for some time. One example is the concept known as the "hired to invent test".[51] If someone is hired to develop a product for a company as opposed to being hired to do something else, the company probably owns the rights to that product.[52] This concept is probably relevant to ownership of software. If a software developer is hired to develop code for a company, in all probability, the company owns the rights to that code. However, if an accountant working for the company develops a package on his own time and has not signed an agreement to the contrary, the accountant probably owns the rights to that software.[53]

Again, employment agreements can help resolve this issue. An interesting case illustrates this concept. In Structural Dynamics Corp. v. Engineering Mechanics Research Corp., three employees, Kothawala, Surana, and Hildebrand, left Structural Dynamics Corp. (SDC)) to work for themselves as EMRC. Before they left SDC, they signed an employment contract agreeing not to divulge SDC's trade secrets and not to compete with SDC for one year after termination. While they were working for SDC, they were developing an isoparametric program for American Motors Corp. Unbeknownst to SDC, after they left they developed this same package for EMRC with some enhancements. AMS purchased the EMRC package, and SDC sued EMRC for violation of the trade secrets and the noncompete clauses. The court disregarded the "hired to invent concept," and dismissed that part of the suit, even though the defendants clearly were employed by SDC to develop the isoparametric software. However, the court did find for SDC on the basis of the noncompete clause and awarded SDC fifteen percent of EMRC's gross revenues from the program and a license fee of $45,000 for the use of its programs.[54] In another example, three employees working for Genasys Corporation bid against their employer on a Small Business Administration contract. They were already working with SBA on another contract, which was coming up for a re-bid. Knowing their employer's labor rates, they secretly bid against him and undercut his rates. They were awarded the contract and then promptly resigned from Genasys. The scheme fell apart when the president of Genasys discovered that the rival company's address was the address of one of the ex-employee's girlfriend. the court imposed a stop-work injunction on the grounds of trade secret theft and violation of the noncompete agreement that each of them had signed upon joining Genasys.[55]

Software developers, in addition to having very definite employment contracts, also protect their trade secrets/commercial software by prosecuting software pirates and other vendors who develop software too similar to theirs. One concept that has sprung from this is "look and feel". These cases involve the similarity of screen displays and icons on screens between two rival packages. Lotus sued Paperback Software because the "Twin" screen display looked almost identical to the "Lotus 1-2-3" screen display. Hewlett Packard and Microsoft were involved in litigation over a similar issue; Apple sued both of them over the use of the "trash can" icon. The issue with "look and feel" becomes, "Who has a right to what?" Does Lotus have an exclusive right to using the F1 key as "Help," or use of slash ("/") to change from data-entry mode to command mode, and no one else?[56] Richard H. Stern, an expert on software piracy and author of The 1984 Chip Protection Act states in an article in

IEEE Micro Magazine: "There should be a public domain for symbols and keys. It is ludicrous and stifles creativity to expect software developers to use different keys to perform the same function in different software packages. I for one do not want to have to learn several different commands in different packages to perform the same function. In fact, I would much prefer to be able to use the same set of commands in each package to save a file, for instance".[57]

Software piracy is a big concern for software developers. Estimates on losses are in the hundreds of millions every year. To help combat this, developers have joined together to form organizations that seek out and prosecute organized software pirates. One organization, Business Software Alliance, recently settled a piracy claim out of court with Singapore-based Computer Systems Advisors. [58] Software piracy in third-world countries is reportedly quite common. The "Super 301" trade bill was passed in part because of lax intellectual property laws in countries like Brazil and India.[59]

LIABILITY WHEN CONFIDENTIAL DATA IS MISUSED

Laws in 1974, 1987 and 1988 were passed to protect individuals from the misuse of confidential / private information (see above). Security standards mandated by the 1987 law clearly require concerns that do business with the federal government to take reasonable measures to protect confidential data to which they have access. In theory, a company that fails to do this and is responsible for exposing confidential data could be held liable for a federal crime. Even before these laws were enacted, however, lawsuits involving similar concepts were litigated.

William Douglas Thompson III applied for a credit card from the San Antonio Retail Merchants Association (Sarma) in 1978. In 1974, William Daniel Thompson opened an account with Sarma that became delinquent. The data-entry operator mistook William Douglas Thompson for William Daniel Thompson and entered his delinquency data into the other Thompson's account. The account names stayed listed as William Daniel Thompson III and contained William Douglas Thompson's social security number. Consequently, the Sarma computer identified the non-delinquent Thompson as a bad credit risk. His numerous attempts to get Sarma to remedy the situation failed, so he finally sued Sarma. The trial court found for the plaintiff and so did the appellate court. The courts noted that Sarma had failed to exercise "reasonable care" in programming the computer to capture the information. Additionally, Sarma used negligent procedures in preparing the consumer report as determined by a "reasonable person" standard. This case illustrates the importance of having conscientious operating procedures to prevent legal liability. The plaintiff was awarded $10,000 plus attorneys fees.[60] This is a very modest amount of money, in light of another case where a man is suing a police force for $10 million in a computer-related mistaken-identity issue.[61]

These cases illustrate part of what a business is required to do under the law, and what can happen when it does not fulfill this obligation. Businesses should make efforts to provide security and privacy of data with respect to individuals. They should ensure that: the system contains only authorized data; only authorized users can access the system; only authorized programs run on the system; only authorized personnel have access to hard copies of the system output. Business management must develop and implement policies regarding personnel practices, system security and physical security.[62]

Under the 1988 law, disclosure of protected information when it is authorized and relevant to the situation is not prohibited, but will require an agreement. In a pre-1988 case, Jaffess v. Secretary of Health Education and Welfare, Jaffess had his veterans benefits reduced as a result of a computer match, which revealed that he had not reported his social-security income as required by law. Jaffess then sued under the Privacy Act of 1974. The court found against Jaffess on the basis that his right to privacy did not extend to intra-agency disclosure of information in the course of conducting regular agency functions. Furthermore, the court noted that the social-security information was relevant to the situation and that the VA was required to use such information when determining how much individuals should receive in benefits.[64] While the court did not find matching to be illegal in this instance, the Computer Matching Law now requires express written authorization for this kind of computer matching.

Computer law is evolving with the technology. Many kinds of crimes can be prosecuted under traditional laws, but new laws have been needed. Currently, the problem lies more with the enforcement of the laws rather than with the lack of relevant laws. Businesses have additional issues to address. They must be concerned with their survival, as well as protecting their investments in information and expertise. The general public should be concerned with who is compiling information about it and how this information in used. The government, law enforcement agencies, businesses, and the general public cannot afford to be ignorant of the legal implications of computer misuse and abuse. Computers and information systems are tools that can greatly benefit society or seriously harm society, depending on the motives of the person using them. As long as the danger of improper information exploitation exists, we must devise appropriate safeguards to ensure the integrity of users and protect the privacy of individuals.

FOOTNOTES

01. Sloan, Irving J., The Computer and the Law, Oceana Publications Inc., 1984, pp. 6-7.

02. Krauss, Leonard I., Computer Fraud and Countermeasures, Prentice Hall Inc., 1979, p. 12.

03. Mandell, Stephen L., Computers, Data Processing, and the Law, West Publishing Company, 1984, pp. 161-162.

04. Sloan, Irving J., The Computer and the law, Oceana Publications Inc., 1984, p. 2.

05. Ibid., p. 3.

06. Ibid., p. 4.

07. Ibid., p. 5.

08. U.S. Department of Justice, Computer Crime Legislative Resource Manual, Koba Associates, Inc., 1980, p. 2.

09. Ibid., pp. 2-3.

10. Ibid., pp. 3-4.

11. Ibid., pp. 4-5.

12. Ibid., p. 5.

13. Ibid., pp. 5-6.

14. Ibid., pp. 7-8.

15. Ibid., p. 8.

16. Ibid., p. 55.

17. Mandell, Stephen L., Computers, Data Processing, and the Law, West Publishing Company, 1984, pp. 163-167.

18. Op cit., p. 9.

19. Ibid., pp. 9-10.

20. Ibid., p. 10.

21. Ibid., p. 11.

22. Ibid., pp. 12-13.

23. Ibid., p. 13.

24. Ibid., pp. 13-14.

25. Wolk, Stuart R., Legal Aspects of Computer Use, Prentice-Hall, 1986, p. 110.

26. Zimmerman, J. S., Computer/Law Journal, Computer Crime Statute, Volume VI, 1986, pp. 461-481.

27. 100th Congress, First Session, Computer Security Act of 1987, Public Law 100-235 [H.R. 145], January 8, 1988, pp. 3120-3125.

28. House Report No. 100-802, Legislative History, Computer Matching Privacy Act, p. 38.

29. Ibid., pp. 38-47.

30. Computer Virus Bill (H.R. 55) Update, Bernard P. Zajac, Jr., Ed., The Computer Law and Security Report, Volume 5, Issue 5, January-February 1990, p. 26.

31. Charles P. Pfleeger, Security in Computing, Prentice-Hall, 1989, p. 494.

32. Zimmerman, J. S., pp. 480-481.

33. Lieberstein, Stanley H.., Who Owns What Is in Your Head?, Hawthorn Books, Inc., 1979, pp. 40-42.

34. Ibid., pp. 40-42.

35. Ibid., p. 110.

36. Ibid., pp. 129-148.

37. Ibid., pp. 36-55.

38. Read in The Washington Post, Business Section, approximately January, 1990.

39. Ibid., pp. B3.

40. Ibid., pp. B3.

41. Lieberstein, Stanley H., pp. 117-118.

42. Ibid., p. 5.

43. Personal Interview with Richard H. Stern, Patent Attorney, on Patenting Software, March 12, 1990.

44. Op city., pp. 59-65.

45. Op cit., March 12, 1990.

46. Lieberstein, Stanley H., pp. 59-65.

47. U.S. Department of Justice, Computer Crime Legislative resource Manual, Koba Associates, Inc., 1980, p. 13.

48. Op cit., p. 1.

49. Ibid., pp. 98-99.

50. Ibid., pp. 42-44.

51. Ibid., pp. 9-10.

52. Ibid., pp. 9-13.

53. Personal Interview with Richard H. Stern, Patent Attorney, on Inventors' Rights to Software, March 14, 1990.

54. Mandell, Stephen L., pp. 96-98.

55. Personal Interview with Gary Smith, Executive Vice President, Genasys Corporation, on Violation of Noncompetition Clauses, March 5, 1990.

56. Stern, Richard H., "Copyright Protection of Screen Displays and Other User Interfaces for Computers: A Problem in Balancing Incentives for Creation Against Need for Free Access to the Utilitarian", IEEE Micro, approximately December, 1989.

57. Ibid.

58. Washington Technology, approximately February 15, p. 2.

59. Ibid., p. 2.

60. Mandell, Stephen L., pp. 190-191.

61. Read in The Washington Post, Section A, approximately January 1990.

62. Op cit., pp. 193-195.

63. Ibid., pp. 188-189.

Applications of Knowledge-Based Systems Techniques to Detect Computer System Intrusions

John McCarron
7608 Barkwood Court
Beltsville, MD 20705

1.0 Introduction

Artificial Intelligence (AI) has had a long history as an academic field of study. Since the early days of this field (circa 1960), experts predicted that there would be very rapid progress toward computers that could outperform people at "difficult" tasks. Until a few years ago, there was much debate as to whether any AI program would ever have any practical use. Although progress has occurred more slowly than anticipated, such debate is dying away nowadays as real and practical artificial intelligence applications have become apparent. The growth in practical applications is due not only to the availability of faster, cheaper computers, but is also (and more importantly) due to the development of easy-to-use knowledge engineering software tools. These tools, called *expert systems*, or *knowledge-based systems*, provide a way to capture and use complex knowledge, or expertise, more quickly and easily than programming the same knowledge in a higher level language such as FORTRAN, C, or PASCAL.

A practical and potentially valuable application area for knowledge-based systems is the intelligent analysis of computer activity to detect intrusions. The human side of computer security management is often the weakest link in the computer security chain. Today's current methods of detecting computer intrusions is not handled well. Even when supplemented by auditing and monitoring functions available in some operating systems, for most practitioners today's technique of manually reviewing system logs and activities for suspicious activity is often done superficially today. A thorough job takes too much time and demands too much from any human's memory or problem solving skills.

This paper will discuss the use of knowledge-based systems to assist a system security officer or system administrator in the intelligent analysis of computer activity to detect intrusions. This paper will be organized as follows:

- Section 2 of this paper will serve as a primer on knowledge-based systems.
- Section 3 will evaluate the tasks involved in computer activity analysis for intrusion detection and discuss the suitability and match of knowledge-based systems to performing these tasks as well as, if not better than, a human.
- Section 4 will survey several actual knowledge-based systems that have been developed over the last few years to provide real-time intrusion detection and to provide analysis tools for the computer security officer to analyze log files and exception reports for intrusions.
- Section 5 will discuss several issues involved in using knowledge-based tools to provide support for computer intrusion detection.
- Finally, Section 6 will present conclusions.

2.0 A Primer on Knowledge-Based Systems

This section will provide some background on knowledge-based systems by defining AI, discussing some basic AI techniques and methods, and finally discussing knowledge-based systems.

2.1 What is AI?

AI has both a scientific goal - to understand intelligence - and an engineering goal of computer applications that exhibit intelligence. AI is seen by some as simply methods for searching for problem solutions, while others believe AI is the study of ideas that enable computers to be intelligent, to the point where computers will be able to outperform people at performing tasks that are "difficult" for humans [13]. The latter definition is more along the lines of the research of applying AI to computer security problems.

2.2 What is a Knowledge-Based. or Expert. System?

A knowledge-based, or expert, system is an AI program that behaves like a human expert in a specialized domain of expertise. These systems gain their power from their knowledge of the domain. Some examples of successful knowledge-based systems include the diagnosis of infectious diseases and configuring of computers.

Developing knowledge-based systems usually involves domain experts and knowledge engineers: a domain expert has knowledge about an area or a task, while a knowledge engineer elicits that knowledge and expresses it in the language used to build the system.

A tool designed specifically for developing a knowledge-based system is an *expert system shell*. The motivation for developing expert system shells was to represent vasts amounts of knowledge or expertise into smaller pieces that most humans could comprehend. These shells are *domain independent*, and provide a superb support environment for the easy development of knowledge-based systems.

The basis of most expert systems is a type of program structure where rules are the basic statements. These type of programs, called production programs, are data-driven programs which consist of:

- a data memory (containing data elements)
- a rule memory (containing condition-action rules)
- an inference engine (to execute or fire rules)

The inference engine continually determines which rules are relevant for existing conditions (as determined by the data), and chooses which ones to fire, or execute. Two kinds of inferencing engines are are usually found in expert system shells: forward reasoning, and backward reasoning. Forward reasoning starts from the current state and infers whatever can be inferred, until no more rules can fire or another rules stops the processing. Backward reasoning starts from a goal and works backwards, selecting rules to fire that may help to reach a goal.

Besides production program capabilities, expert system shells also have features for building user interfaces, communicating with conventional programming languages, and using data outside the expert system. Some shells have facilities for expressing uncertainties and propagating conclusions based on inference. Another important feature of the expert system shell is the ability to generate explanations automatically; people need to know why the system reached a conclusion, what rules it used, and what sequence of reasoning.

2.3 Advantages and Benefits of Developing Knowledge-Based Systems

Knowledge-based systems have advantages over conventional programming languages for problem domains characterized by:

- uncertainty
- lack of structure

- a large unwieldy set of rules that are hard to comprehend
- the need for heuristics (rules of thumb for deduction or decision making)
- few experts in the field

Developing a knowledge-based system can benefit their sponsors in several ways:

- They can provide a way of carrying out a task better or at a lower cost (or both).
- They can multiply scarce expertise which can be used at more than one site.
- They may gather and unify expertise from many sources, such as different experts, books, and other documentation.
- They should be able to play the role of teacher, learner, consultant, and expert problem solver.

3.0 Applicability of Knowledge-Based Systems to the Computer Intrusion Detection Domain

3.1 Weaknesses of Current Computer Activity Analysis Methods

The human aspect of computer security is often the weakest link, due to the following factors:

- Real-time activity analysis for intrusion detection is extremely limited to operating system auditing and monitoring functions; in most cases it's virtually impossible to catch an intruder in real-time.
- Security policy for a computing system is usually specified as a set of access rules, stating who can use what resources and in what manner. The task of a human understanding the combinations of these rules is complex, and therefore the rules are not generally well understood.
- Conventional computer activity analysis is a human intensive task, where the computer security administrator or security auditor spends a great deal of time perusing logs of system activity.
- Even with log files, in most environments it is almost humanly impossible to review all activity. For example, a typical log file of VMS image termination data for 100 users can generate upwards of 20 megabytes of data per week; more detailed collection could result in as much as 1000 megabytes a week for those same users. A thorough job by manual means takes too much time and demands too much from human memory and problem solving abilities to do properly.

3.2 Helpful Ways to Use Knowledge-Based Systems

Knowledge-based systems provide a way to capture and use complex knowledge, or expertise, more efficiently than programming in a high-level procedural language. The value of these systems lie in the inherent efficiency of customizing the use of facts or rules of the knowledge base to a specific current situation in a way that can't be done by a passive method like consulting a book or manual. Information is supplied about 'what to do', 'when to do', and 'when not to do'. Preston [10] suggests that knowledge-based systems have proven helpful when used in several of the following ways:

- As a human expert's assistant, functioning as a super checklist of diagnostic tests to run, or ranking diagnoses in order of probability. Since the software never gets in a hurry, and never gets tired or forgetful, it can achieve a high quality of uniformly good results within a well-defined application domain.
- As an expert within a well specified application domain, some systems have produced results consistently equal to, or better than, human expertise by combining the knowl-

edge of more than one expert. Such systems can provide a source of scarce and expensive expertise to a trained, but non-expert user.

- As a monitoring and control system, where control in some cases can be both better and faster than a human operator.
- As a replacement or suppliment to a procedures manual. The expertise lies in providing the correct selective access to the procedures.

3.3 Task Suitability

Experienced builders of knowledge-based systems have suggested guidelines to determine if a particular task is a good candidate for such a system. Guidelines from several sources indicate that computer security, and especially computer activity analysis to detect system intrusions, seems to be a very good match for the use of

Three suggestions from Hayes-Roth et al.[4] are the following:

- The task should be a narrow specialty that does not involve much common sense knowledge.
- The task should take a human expert more than a few minutes, and less than a few months, and have no more than a few hundred relevant concepts.
- The nature of the inputs and outputs should be desirable fairly precisely, and there should be access to many specific examples of problems and solutions.

Slague and Wick suggest a more formal method of evaluation [4]. Several of the characteristics listed as essential features of success are:

- The task is not natural language intensive.
- The task is knowledge intensive.
- The task may be heuristic, and would be hard or impossible to perform by an algorithmic approach
- Both hard and easy test cases are available.

Based on these characteristics of a good knowledge-based system, the weaknesses of current techniques to analyze system activity, and the strengths of knowledge-based systems where conventional techniques are weak, it seems rather obvious that creating a knowledge-based system to aid in computer activity analysis and intrusion detection would be a good idea. With the increasing availability of expert system shells and faster computers (all at reasonable prices) this application area seems ideal. The next section surveys knowledge-based intrusion detection systems that have been developed over the last few years.

4.0 Survey of Current Knowledge-Based Computer Intrusion Detection Systems

4.1 Denning's Framework for an Intrusion Detection Model

A model of intrusion detection, described by Denning [3], provides a framework for an intrusion detection expert system. Denning states that the exploitation of a system's vulnerabilities lies in the abnormal use of the system; therefore security violations can be detected from abnormal patterns of system use. The following types of threats could be detected by thorough computer activity analysis:

- Attempted Break-In
- Masquerading, or successful break-in

- Penetration by a legitimate user/misuse of privileges
- Leakage by a legitimate user
- Inference by a legitimate user
- Trojan Horse
- Virus
- Denial of service

The basic idea is to monitor the standard operations on the target system, such as logins, command and program executions, file and device accesses, etc., looking only for deviations in usage. Statistical measures are used to determine what is normal and what is abnormal use.The six main components to Denning's intrusion detection expert system are subjects, objects, audit records, profiles, anomaly records, and activity rules.

Subjects are initiators of activity on a target system, who are normally users. A subject is typically a terminal user, but it might also be a process acting on the behalf of users or groups of users, or it might be the system itself. Subjects may be grouped into classes for the purpose of controlling access to objects in the system.

Objects are resources managed by the system, such as files, commands, devices, etc. Objects are grouped into classes by type (program, text file, etc.).

Audit records are generated by the target system in response to actions performed or attempted by subjects on objects, such as user login, command execution, file access, etc. This time-stamped record identifies the subject, object, and action, and also describes any exception condition generated and resources used, such as CPU time, number of records read, etc.

Profiles are structures that characterize the normal behavior of subjects with respect to objects in terms of statistical metrics and models of observed activity. New observations are considered abnormal if they fall outside confidence intervals established by the statistical models.

Anomaly records are generated when abnormal behavior is detected in the target system.

Activity rules are the actions taken when an audit record or anomaly record is generated, or when a time period ends. Activity rules consist of two parts: a *data condition*, which when satisfied) causes a rule to be fired (or executed), and the *actions* to be taken. The condition is specified as a pattern match to an event.

The model is basically a rule-based pattern matching system. When an audit record is generated, it is matched against the profiles. Information in the matching profiles then determines what rules to apply to update the profiles, check for abnormal behavior, and report anomalies detected. The security officer assists in establishing profile templates for the activities to monitor.

This model does not contain any special features for dealing with complex actions that that exploit a known or suspected security flaw in a target system; as a matter of fact, it has no knowledge of the target system's security mechanisms or its deficiencies. A flaw-based detection system may have some value. but it would be very difficult to build and complex to maintain. By detecting an intrusion however, the security officer may be able to locate vulnerabilities.

The goal of Denning's work is real-time intrusion detection, which would allow a system to respond rapidly before serious damage is done. The same model can be used to analyze log files at a later time.

4.2 Systems Developed Using Denning's Model

Several intrusion detection expert systems have been developed using Denning intrusion detection model as a framework the system Some of these systems in will be described in this section.

4.2. 1 IDES

A team at SRI International is developing a prototype real-time intrusion detection expert system based on Denning's intrusion-detection model. This system, called IDES [6-8], is an independent system that runs on its own hardware and processes audit data characterizing subject activity received from a target system. The goal of IDES is to provide a system-independent mechanism for real-time detection of security violations, whether they are initiated by outsiders who attempt to break into a system, or insiders who attempts to misuse their privileges.

The developers of IDES found some weaknesses and difficulties attempting to detect intrusions solely on the basis of departures from observed norms for individual users:

- Although some users have well-established patterns of behavior, others may have erratic work hours, may differ radically day to day in the amount and type of their activity, and may use a computer from several different locations or even time zones (e.g., at home, at the office, on travel). For erratic users, almost anything may be considered "normal", and an intruder might easily go undetected.
- A statistical approach may not be so successful with with legitimate users who abuse their privileges, especially if such abuse is "normal" for those users.
- The approach is vulnerable to defeat by insiders who know that their behavior is being compared to a previously established behavior pattern. These users can slowly vary their behavior over time, until they establish a new behavior pattern which can be used for intrusions and system attacks.

The IDES system incorporates a second line of defense against these weaknesses by also incorporating rules that describe suspicious behavior that is independent of past behavior patterns. This second part of the expert system rulebase will encode operating system dependent and installation specific rules about:

- intruder behavior
- known system vulnerabilities
- reported attack scenarios
- intuition on suspected behavior

The statistical profile rules coupled with system and installation specific intrusion detection rules gives IDES the potential to be a strong intrusion-detection system. In preliminary experiments, IDES has been capable of detecting and reporting anomalous system behavior in real-time. Work is still continuing on this project.

4.2.2 NIDX

Researchers at Bellcore have developed a software system called Network Intrusion Detection Expert System NIDX) as part of efforts to address network security issues [2]. NIDX is a knowledge-based system that will examine a real-time, audit trail of user activity on a computer system to detect security violations. NIDX is being designed to combine knowledge describing the target system and security related heuristics to form a knowledge-based system capable of detecting specific violations that are potential threats to the target system. Current efforts are concentrated on the UNIX System 5 environment, but the same concepts can be extended to other systems.

The NIDX model is based on Denning's framework for an intrusion detection model, but it has been extended to include system-dependent knowledge to allow detection of specific security violations from the target system audit trail. An example of system dependent knowledge is a description of the target system's file system and heuristic - and policy- based rules for detecting violations. Besides the six basic components of Denning's model, four other components are also used in the NIDX system:

- *Intrusions*: the security violations that the UNIX intrusion detection system detects (e.g., unauthorized browsing, modification, information theft, etc.)
- *Suspicious events*: events that individually are not intrusions, but may be an indication of an impending security violation (e.g., user login exceptions)
- *Knowledge Base*: contains knowledge about the system being monitored, and heuristics for detecting intrusions
- *Rules*: used to update session profiles from audit records, detect suspicious events from audit records, and detect activities which are indications of intrusions.

The goal of the NIDX model is to support the development of a system that can characterize and detect specific intrusions from an audit trail of user activity. In the NIDX approach, intrusions are detected via a two-step process. First, the audit records of user interaction with system objects are analyzed and placed in one of the following active categories: browse, modify, delete, or copy. Then the activity is authorized according to policies and heuristics contained in the knowledge base.

Thus, in order to detect unauthorized browsing, the system first detects that the user is browsing, then determines if the activity is an intrusion. Detecting browsing requires NIDX to know the semantics of the audit trail records. Also, determining whether a detected activity is an intrusion requires NIDX to have access to policies and heuristics describing valid/invalid user behavior.

The current NIDX model is appropriate for network environments where formal and informal policies exist to allow differentiation between authorized and unauthorized behavior. A key objective of the real-time analysis is to allow real-time containment of an intruder's activity, limiting the damage that can be done. NIDX is currently in the prototype stage of development.

4.2.3 Discovery

The Discovery expert system [13] is based on an approach similar to Denning. Discovery was developed for TRW Information Services, where the users are subscribers to a service, and where dial-up access is the norm. Some of the security measures used in more controlled situations are thus not available. Discovery develops (and updates daily) profiles of normal subscriber activity, then compares current activity with these profiles to detect potential violations. These are then further analyzed by human investigators. However, no details are available on the pattern recognition methods.

4.3 Other Detection Intrusion Models

Other knowledge-based intrusion detection models using an approach different form Denning's are currently under investigation. This section will briefly describe some of those models.

4.3.1 Wisdom and Sense

A real-time anomaly detection program called Wisdom and Sense has been developed as part of a real-time intruder detection system by researchers at Los Alamos National Laboratory [S] . This model based on representing system user behavior by a non-trivial statistical model, then automatically developing a rulebase for detecting anomalous system use based on this data.

The goal of all anomaly detection modules is to identify the set of least frequent transactions such that the sum of the total expected frequency is less than an small value (say 1%). In practice, the rules that enable this identification of transactions need to be generated from a small sample of all possible transactions. This is a non-trivial task. Typically, anomaly detection rules are generated based on one-millionth to one-billionth of all possible transactions. But even then, this small sample might consist of several thousand rules!

A further complicating factor is the types of data that can be gathered to detect anomalies. Due to the nature of the data that can be gathered to detect anomalies, anomaly detection can not be implemented by simply checking how frequently a transaction has taken place in a historical database or by classical parametric statistical estimation approaches.

In the Wisdom and Sense approach, the metrics of interest are represented by a nonparametric density estimation approach. Then, based on historical observations, a rulebase can be automatically generated. These rules specify legal values (commonly observed values legal to others) of "test fields" conditioned on the values of one or more other fields. For example, a generated rule might state that if the user is "gunar", the time is between 8:00 am and 5:00 pm, and the day is Wednesday, then the legal ports are portl and port2. If a transaction occurs such that "gunar" logs into the system on port3 at 12:20pm on Wednesday, the transaction violates the previous rule, and therefore the rule contributes some evidence that this is an anomaly. In this way, Wisdom and Sense can be thought to extrapolate the available information of what values combinations can be expected to be common and which are expected to be unusual. Anomalies can then be reported to the security officer immediately, rather than reading megabytes of data from a log file and "guesstimating" anomalies.

Efforts are continuing to further establish the soundness of this approach through rigorous analysis. Wisdom and Sense is currently under beta testing at several test sites.

4.3.2 Halme and Van Home Model

Halme and Van Home have researched knowledges-based intrusion detection systems for Sytek [13]. Their experiments were conducted in a UN˜ environment. An automatic pattern classification technique was developed via an expert system to identify features that successfully discriminate intrusion sessions from normal sessions. Examples of such features used in this pattern recognition technique include password changes, queries of user identities, access to system dictionary, and new device privileges for users.

5.0 Limitations and Issues

There are several open questions concerning the use of knowledge-based systems for intrusion detection Some of those issues will be discussed in this section:

5.1 Limitations of Knowledge-Based Systems

Current knowledge-based systems do have some limitations that must be addressed. Some limitations are the following:

- Today's working systems are not very capable of learning and independently improving performance over time (as a human does).
- Knowledge-based systems sometimes lack recognition that a problem is outside area of expertise, which can then give grossly mistaken and unreasonable answers.
- Run-time/ performance limitations.
- Some types of knowledge are very hard to represent in current tools, and some types of reasoning (especially when 'facts' change after they are asserted) are not available in all tools.

5.2 Limitations and Issues for Current Knowledge-Based Intrusion Detection Systems

Most of the knowledge-based intrusion detection systems are still in the initial research phase. Most of the Models are based on Denning's approach, with some obvious enhancements. Denning has raised several issues about using knowledge-based systems for intrusion detection [3]:

- *Soundness of Approach* . Do these approaches actually detect intrusions? Can one readily detect anomalies related to intrusions from those related to other factors ? Most of the known systems are in the prototype stages with a lot of development work to go before become reliable commercial systems.
- *Completeness of Approach*. Does the approach detect most, if not all, intrusions, or is a significant proportion of intrusions undetectable by these methods? Many feel that the combination of a statistical approach and an expert system approach within one intrusion detection system is a very very comprehensive.
- *Choice of Metrics*. What metrics, models, profiles, and rules provide the best discriminating power? Which are effective? What are the relations between anomalies and intrusions?
- *Criterion for Success*. Since real intrusions can not be easily manipulated experimentally, how are errors measured? One person even suggests that expert systems should be created to simulate a system intruder, and thus test the intrusion detection system [10].
- *Feedback*. What effect should detection of an intrusion have on a target system ? Should these systems automatically take certain action?

5.3 Privacy Issues

The issue of privacy in the use of computer security has been raised by several authors. These authors have voiced concerns about the use of computer monitoring for computer security purposes, and have suggested that security measures such as intrusion-detection mechanisms may actually increase the threat of computer abuse by engendering employee dissatisfaction [3]; whence the emergence of the disgruntled employee and the so-called insider threat. It has been suggested that even the knowledge of the existence of an intrusion detection system would be an aid to a would be intruder [8].

Intrusion detection systems make possible an even greater degree of invasion of privacy and other potential objectionable activity, such as employee performance monitoring. However, even in benign security mechanisms such as file backups, archives, and keeping audit trails of computer activity, there is great potential for abuse of such data. These points only emphasize the need for concern for the appropriate use of intrusion detection technology. Such system must be used in a reasonable manner to be effective at enhancing computer security.

6.0 Conclusions

Conventional computer security tools provide various levels of protection against potential intruders. However, this protection is given only as long as the preventive mechanisms are not compromised. Once these mechanisms are defeated by a sufficiently determined intruder, the underlying system is again left open to tampering that may be difficult (if not impossible) to trace by conventional means.

A complimentary mechanism to intrusion prevention is detection. Conventionally, this is achieved by providing detailed audit data that may allow security officers to detect intrusions, assess damage, and then repair security holes. However, this human side of computer security management is often the weakest link in the computer security chain. Today's current methods of detecting computer intrusions are not handled well. Even when supplemented by auditing and monitoring functions available in some operating systems, for most practitioners today's technique of manually reviewing system logs and activities for suspicious activity is often done superficially today. A thorough job takes too much time and demands too much from any human's memory or problem solving skills.

Knowledge-based techniques appear highly promising as the basis of tools for computer system intrusion detection. By building system statistical usage profiles, incorporating rules for system security policy, and developing heuristics for detecting intruder behavior within a target system environment, a knowledge-based intrusion detection system can analyze the system audit trail in real-time to detect intruder activity.

However, one must note caution. Based on the currently implemented and/or research versions of these systems, much work is needed before the techniques can be widely applied. There have been relatively few working systems; those few have required large and consistent investment beyond what many users could justify in a rational benefit cost proposal. However, in time and with increased research commitment, a practical and 'real' application of artificial intelligence may be on the horizon for the computer security masses.

References

1. Baldwin, R.W., "Rule Based Analysis of Computer Security", MIT Laboratory for Computer Science, 1987.
2. Bauer, D.S., and Koblentz, M.E.,"NIDX - A Real-Time Intrusion Detection Expert System", Proceedings of Summer USENIX Conference (1988), pp. 261-273.
3. Denning, D.E., "An Intrusion Detection Model", IEEE Transactions on Software Engineering, Vol. SE-13, No. 2 (1987), pp. 222-232.
4. Hayes-Roth, F., Waterman, D.A., and Lanat, D.B., Building Expert Systems, Addison-Wesley, 1983.
5. Liepins, G.E., and Vaccaro, H.S., "Anomaly Detection: Purpose and Framework", Proceedings of the 12th National Computer Security Conference (1989), pp. 495-504.
6. Lunt, T.F., "Real-Time Intrusion Detection", Proceedings of the IEEE Symposium on Security and Privacy (1989), pp. 348-353.
7. Lunt, T.F., and Jagannathan, R., "A Prototype Real-Time Intrusion-Detection System", Proceedings of the IEEE Symposium on Security and Privacy (1988), pp. 59-66.
8. Lunt, T.F., et al., "Knowledge-Based Intrusion Detection", Computer Science Laboratory, SRI International, 1987.
9. Pfleeger, C.P., Security in Computing, Prentice Hall, Englewood Cliffs, 1989.
10. Preston, C.E., ''Artificial Intelligence Applied to Information System Security", Information Age, Vol. 11, No. 4 (1989), pp. 217-224.
11. Rich, E., Artificial Intelligence, McGraw-Hill, New York, 1983.
12. Slague, J.R., and Wick, M.R., "A Method for Evaluating Candidate Expert Systems", AI Magazine, Vol. 9, No. 4 (1988), pp. 44-53.
13. Summers, R.C., and Kurzban, S.A., "Potential Applications of Knowledge-Based Methods to Computer Security", Computers and Security, Vol. 7, No. 4 (1988), pp. 373-385.

THE DESIGN OF THE TRUSTED WORKSTATION: A TRUE "INFOSEC" PRODUCT

Frank L. Mayer[1]
J. Noelle McAuliffe

Trusted Information Systems, Inc
3060 Washington Road (Route 97)
Glenwood, MD 21738

ABSTRACT

In recent years it has been recognized that the protection of classified and sensitive information in an distributed automated processing environment requires a total "information security" (INFOSEC) solution, combining both communications and computer security technologies into an integrated security solution. While the need for INFOSEC solutions is clearly recognized, the commercial availability of true INFOSEC products is extremely limited, or non-existent. This paper discusses the results of an effort to take commercially available COMSEC technology and commercially available trusted system technology, and integrate them into a readily available and evaluated INFOSEC product.

1. INTRODUCTION

The proliferation of commercially available products for the protection of sensitive and classified information is becoming a reality with the successes of the National Security Agency (NSA) Commercial COMSEC Endorsement Program (CCEP) and the National Computer Security Center (NCSC) program to evaluate commercially available trusted systems. In recent years, these security communities have come to realize that both arms of the *information security* (INFOSEC) problem, i.e., communication security (COMSEC) and computer security (COMPUSEC), are necessary to ensure the complete protection of sensitive information. However, the commercial availability of true INFOSEC products is nearly non-existent, despite the successes of the COMSEC and COMPUSEC halves of the problem. This paper discusses a product under development at Trusted Information Systems, Inc. (TIS) that makes available an INFOSEC product that is based upon previously approved and commercially available COMSEC and trusted system products. Specifically, this product, the TIS Trusted WorkStation (TWS), incorporates a NSA approved personal computer encryption device (PCED) into a trusted system environment.

2. BACKGROUND

The major component of the PCED is an add-on board for an IBM PC compatible computer architecture. The PCED system also includes a plain/cipher switch, an interface for a key loader device, and an RS-232 asynchronous communications port (see Figure 1). The plain/cipher switch provides a user with the ability to determine whether data leaving the computer is encrypted. The PCED must be keyed with paper tape keying material via the key loader before being used for communication. The communication software approved for the

[1] Mr. Mayer's current address is SPARTA, 9861 Broken Land Parkway, Columbia, Maryland 21046.

PCED is written for the DOS operating system. The PCED is designed to allow users, utilizing ordinary personal computers, to communicate classified information over non-secure communication channels.

The general scenario for operation from the user perspective is:

1. Key the PCED with keying material approved for the level of classified data to be transmitted.

2. While in plain text mode, establish the necessary communication channel (e.g., direct modem-modem connection, via DDN or other wide-area net).

3. When ready to transmit classified information, switch the PCED into cipher mode and send information.

4. Return to plain text mode and discontinue communication.

In essence, the PCED is designed as a cryptographic guard that sits between a personal computer processing classified data and an unprotected communications line.

Logically, two data streams, plain and cipher, are handled by the PCED. On output from the computer, data sent to the PCED is encrypted (or not) before transmission based upon the position of the plain/cipher switch. On input to the computer, data received by the PCED is decrypted (or not) depending upon the contents of the data itself. In other words, decryption is data driven whereas encryption is determined manually via the plain/cipher switch. Between the communication software on the computer and the PCED, the plain and cipher data streams are interleaved with each other on a byte by byte basis. In the DOS version of the PCED system, the application program, along with the user, is responsible for separating these interleaved data streams (see Figure 2).

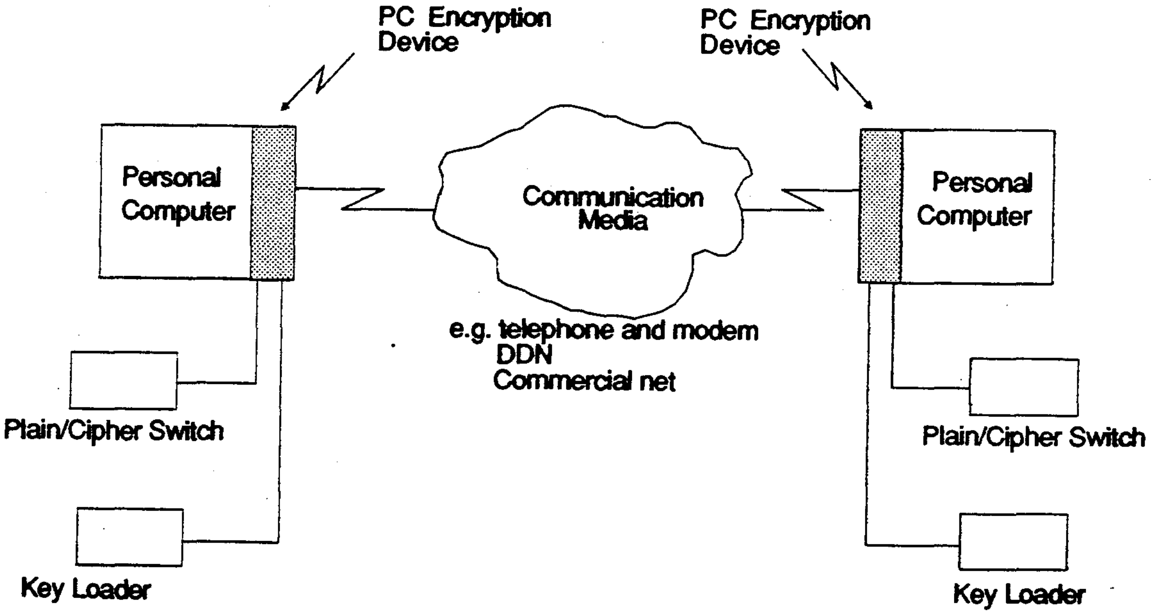


Figure 1
Overview of the PCED System

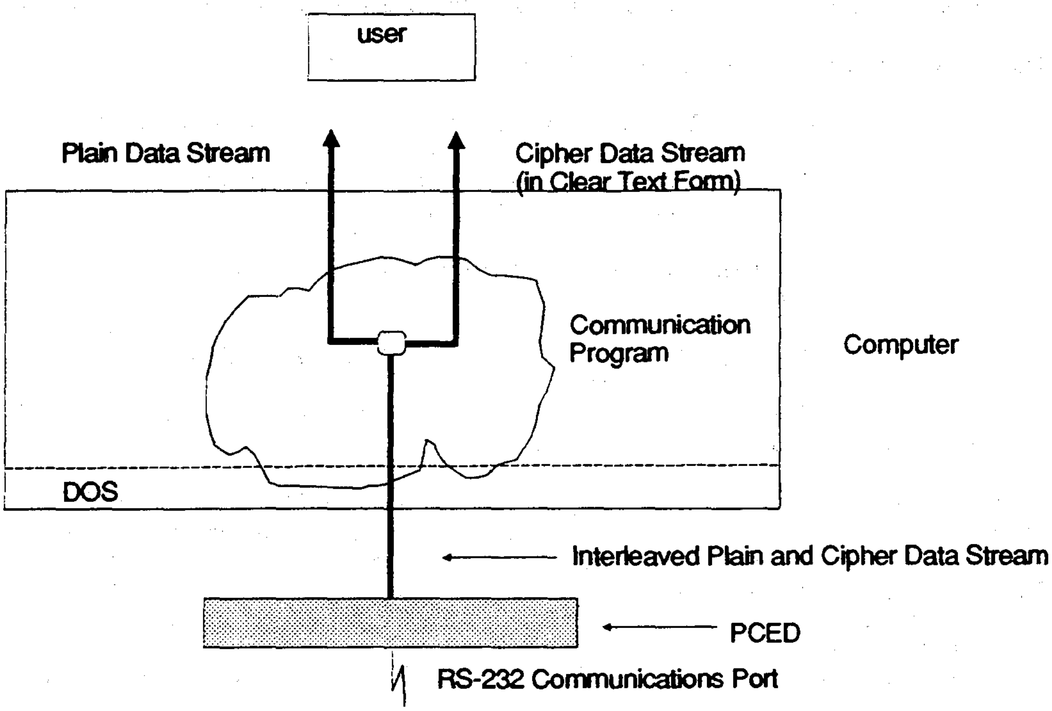

user
Plain Data Stream
Cipher Data Stream (in Clear Text Form)
Communication Program
Computer
DOS
Interleaved Plain and Cipher Data Stream
PCED
RS-232 Communications Port

Figure 2
The PCED device in a DOS Environment

2.1 The Problem

The problem with using DOS to support the PCED system is that DOS lacks any real computer security controls, allowing for opportunities where classified information can be disclosed in the clear, either accidentally or intentionally. For example, a user can forget to switch the PCED into cipher mode before transmitting classified data, thereby sending the data in the clear. More important, any of the known programming attacks (e.g., Trojan Horses, viruses, worms) can easily ignore the intended use of the PCED and transmit information freely in the clear, or spoof the user into believing that data received was plain text data when in actuality it was (presumably more sensitive) cipher text data. It is entirely feasible that any such attack could illicitly "leak" classified information without the knowledge of any human user.

The PCED system is illustrative of the more general problem that an entirely COMSEC (or for that matter, an entirely computer security) protection philosophy is insufficient to ensure the total protection of classified or sensitive information.

2.2 The Solution

In distributed processing environments the total protection of classified or sensitive information can only be assured through the proper utilization of both communication and computer security technologies. In the PCED specific case, where in essence the computer is handling multiple levels of data sensitivities (i.e., classified data and an unclassified communications port), it follows from [1, 2] that a B2 or higher trusted operating system as defined by the *Trusted Computer System Evaluation Criteria* (TCSEC) [3] is necessary in order to assure appropriate control of the PCED. With this understanding, TIS, in early 1988, began an internally-funded effort to integrate available COMSEC technology (i.e., PCED) with available trusted system technology into an INFOSEC product called the Trusted WorkStation (TWS). For the TWS, IBM[2]'s Secure Xenix[3] [4] operating

[2] IBM is a registered trademark of the International Business Machine Corporation.

system [4] was chosen as the trusted operating system because of its targeted B2 rating, commercial availability, and support for a hardware architecture compatible with the PCED (i.e., the PC AT).

The intent of the TWS is to demonstrate that COMSEC and trusted systems technologies can be integrated into a fully functionally INFOSEC product that is available today, using today's best available technology, in a relatively short period of time. It is our intent to have the TWS product re-evaluated by both the communication and computer security organizations within the National Security Agency to ensure that neither the PCED nor the Secure Xenix evaluations have been invalidated by the TWS integration effort.

3. TWS SYSTEM DESIGN

The primary objective for integrating the PCED within a trusted system environment is to prevent unauthorized or inappropriate disclosure of sensitive or classified information by untrusted application software via the PCED communications port. Since the purpose of the PCED is to separate data of differing sensitivities, it is necessary to ensure that high level data is not transmitted in the clear and that incoming decrypted data (which supposedly was high level data generated by another workstation) is not mishandled as low level data. Such an environment is in effect a multilevel secure mode of operation which would require the type of features and assurances provided by a B2 or higher trusted system.

The basic security policy for the TWS is centered on the notion that the PCED manages two logically separate *data streams*--cipher text and plain text--each of which have different associated sensitivities. To accommodate this security policy view, the protection mechanisms introduced into Secure Xenix abstract the PCED into two distinct objects, called PLAIN and CIPHER. Hence, the philosophy of protection is simple; each object is labeled with its sensitivity and access to these objects is determined in a fashion comparable to other Secure Xenix objects.

The labels for both the PLAIN and CIPHER objects are site settable, allowing the TWS to be configured into many different operational environments. As with all devices in Secure Xenix, the PCED has an associated maximum and minimum security level, referred to as *g_low* and *g_high*. By definition, the level of the PLAIN object is the same as that defined by the security administrator for g_low. The presumption is that g_low reflects the level of protection afforded the communication channel attached to the PCED's RS-232 external connection, and therefore the sensitivity of all plain text data sent or received via the device. For example, if the PCED is used to communicate via a commercial telephone system, g_low (and therefore the level of the PLAIN object) would be unclassified. The security level for the CIPHER object is set to the level of the current encryption key, which is loaded by an authorized user. The presumption here is that the current key determines the security level at which the PCED, in cipher mode, is approved to protect information from unauthorized disclosure. The TWS security policy requires that the key level be within the inclusive range of security levels defined by g_low and g_high.

The remainder of this section discusses the architectural and design issues with respect to integrating the PCED and Secure Xenix to implement this protection philosophy.

3.1 Overview of Secure Xenix

The IBM Secure Xenix system is a re-implementation of the Xenix operating system to accommodate the B2 requirements [5]. The major changes to the system are the addition of more sophisticated protection and security policy features, and a restructuring of the Xenix kernel to comply with the B2 system architecture requirements. Some of the most significant changes to the system are in the areas of access controls. The primary means of access control within Xenix is protection bits associated with every file, which provide for discretionary access control at a user, group, and world level of granularity. For Secure Xenix, IBM extended this notion of discretionary access control to a fully generalized access control list (ACL). Additionally, to accommodate the B2 requirements for mandatory access controls, IBM introduced process and object labeling, and the associated

[3] Xenix is a trademark of Microsoft Corporation.

[4] When this effort was originally conducted, IBM was still marketing and supporting Secure Xenix. At the time of this writing, TIS has acquired Secure Xenix from IBM and has renamed the product Trusted Xenix.

access control restrictions that use these labels [6]. Other significant changes to Xenix include more sophisticated system administration support, including the near elimination of the overly powerful Xenix "superuser". For a greater understanding of the Secure Xenix architecture, see [5].

The Secure Xenix system is currently commercially available and has completed developmental evaluation with the NCSC in December 1987. The system is now under formal evaluation at the NCSC with a targeted evaluation class of B2 [7].

3.2 TWS Extensions to Secure Xenix

The most significant TWS extension to Secure Xenix is the introduction of software into the kernel that directly interacts with the PCED hardware. This software is where the majority of additional access control mechanisms are implemented. The TWS also includes a number of trusted, non-kernel support programs to facilitate user management of the PCED in a reliable manner. Additional TWS components include trusted software that provides a more flexible programming interface to the PCED and untrusted user application programs that facilitate communication via the PCED. Each of the major components of the TWS system are discussed below.

3.2.1 PCED Device Driver

Within Secure Xenix, user processes interact with hardware devices through a set of kernel routines called device drivers. A device driver is accessed via the Xenix file system using a special form of a file called a *device special file*. Specifically, the PCED driver is represented by the character device special file "/dev/gi". The major responsibility of the PCED driver is to separate the plain and cipher data stream, and present them as the two logically distinct objects, PLAIN and CIPHER, thus removing the ability of untrusted application software from inappropriately disclosing data via the PCED (see Figure 3).

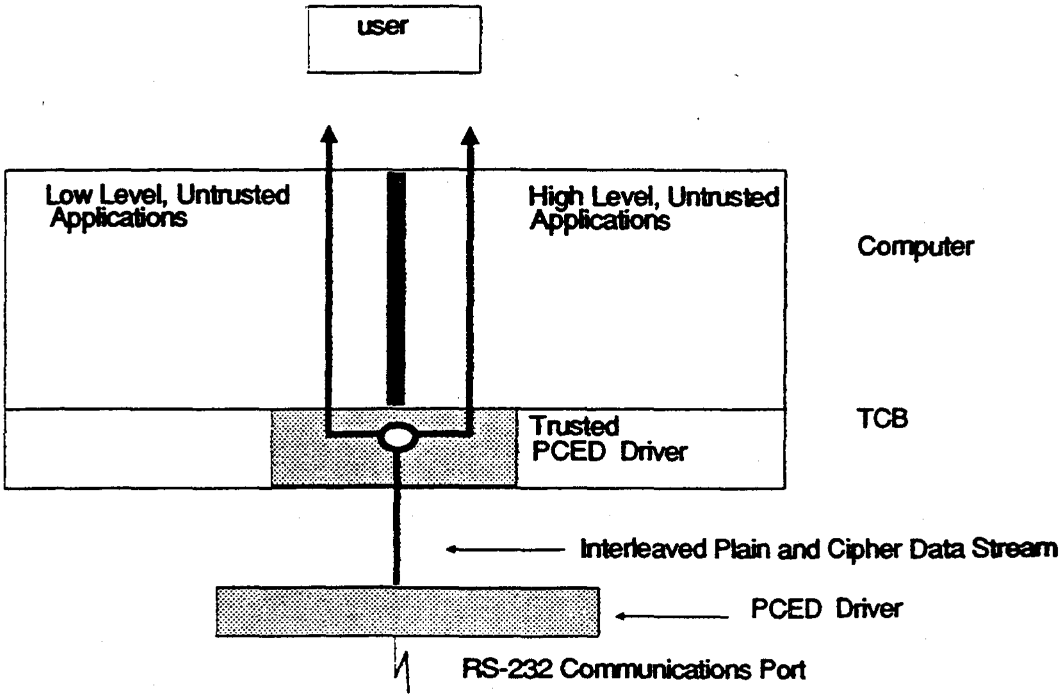

user
Low Level, Untrusted Applications
High Level, Untrusted Applications
Computer
TCB
Trusted PCED Driver
Interleaved Plain and Cipher Data Stream
PCED Driver
RS-232 Communications Port

Figure 3
PCED Integrated into the Secure Xenix System

The Secure Xenix kernel mediates access to all files, including device special files, when a process attempts to "open" the file or device. As with ordinary files, all device special files are assigned a security level and access permission bits, which the kernel uses in its mandatory and discretionary security policy decisions. This type of security policy follows from the notion that a typical Xenix device, such as a terminal or a communications port, is a single object with a single set of security attributes. However, as previously discussed, the device driver actually presents the PCED as two separate objects, PLAIN and CIPHER, each of which has a potentially different associated sensitivity level. Since Secure Xenix provides for only a single security level for every device, and uses this level accordingly for mandatory access control decisions, it was necessary for the TWS to add additional mandatory controls within the driver to accommodate a single device special file that represents two separate security objects. For this reason the PCED special file has a "wildcard" security level, which allows any process, regardless of current security level, to pass the kernel's mandatory access controls and reach the driver's routines on an open call. This use of a "wildcard" security level passes the burden of mandatory access mediation for the PLAIN and CIPHER objects onto the PCED driver itself.

The PCED driver supports the typical Xenix file system operations (e.g., open, close, read, write) plus device specific operations (e.g., load a cryptographic key, set communications parameters, set baud rate). The PCED driver's *open* routine is where access mediation is performed within the PCED driver. The *open* routine will allow a calling process to access the device if the device is not currently opened by another process. To facilitate this locking, the driver maintains a "busy" flag which is set whenever the device is successfully opened and reset when the device is closed[5]. Once the device is locked by a calling process, access mediation to the CIPHER and PLAIN objects is performed. If a cryptographic key has not been previously loaded, the calling process is awarded no access to either object, but may keep the device opened for other purposes (e.g., loading a key). In performing access mediation, the *open* routine attempts to provide the calling process as much access as allowable under the system security policy to both the PLAIN and CIPHER objects. The access control rules enforced by the driver's *open* routine are essentially as follows:

(1) If the level of the calling process *dominates*[6] the level of the PLAIN object (which is g_low), then the calling process may "read" the PLAIN object.

(2) If the level of the calling process *dominates* the level of the CIPHER object (which is determined by the key level), then the calling process may "read" the CIPHER object.

(3) If the level of the PLAIN object *dominates* the level of the calling process, then the calling process may "write" the PLAIN object.

(4) If the level of the CIPHER object *dominates* the level of the calling process, then the calling process may write the CIPHER object.

A simplified view of this mediation is presented in Table 1. The *open* routine also audits the access it grants to the calling process.

Once the device is successfully opened, the calling process may perform read and write operations as allowed by the driver's access control rules. A side-effect of this approach to access control is that some data received by the PCED may be lost if the process for which the device is currently opened is not authorized to read information as it is received. This side effect occurs because the driver does no buffering of the plain and cipher data streams. For example, if a process, with the device opened such that it may read the PLAIN object but not the CIPHER object, issues a read request to the driver and the next available byte of data from the interleaved data stream is cipher data, the byte is lost and the process is told that no data was available (to avoid a potential covert channel).

The PCED driver also supports a set of ancillary system calls that facilitate management of the PCED. Two of these calls are privileged operations. The first allows a process to load a *message indicator* (MI) into the device. The MI is used in the encryption and decryption process, and is typically loaded only once during

[5] The "busy" flag introduces a known covert channel, since processes at multiple levels can attempt to open the device. This covert channel is handled through bandwidth limitation techniques.

[6] "Dominates" here refers to the *dominance* relation as defined by the Bell-La Padula model [8, 9, 10].

Table 1
Simplified View of PCED Driver Access Mediation

Process Level	Allowed Process Access PLAIN object (LOW)	 CIPHER object (HIGH)
LOW	Read, Write	Write
MID	Read	Write
HIGH	Read	Read, Write

The above example assumes g_low = LOW and the key level = HIGH and a strict ordering of security levels (i.e., no categories) such that:

HIGH *dominates* MID *dominates* LOW.

installation of the PCED board. The second privileged call provides for the loading of the cryptographic key. Any number of keys may be used by the device (though only one at a time), thus this call is a much more frequently used operation. At a minimum, a key must be loaded after every system initialization since during system start up, the currently loaded key is automatically destroyed. The key load driver routine also accepts, as input parameters, the security levels for the PLAIN (which is g_low) and the CIPHER (which is the key level) objects. Only trusted, non-kernel processes (which are described in the next section) may use these two privileged calls.

The remaining ancillary calls supported by the PCED driver provide for setting communication parameters, querying current security levels of the PLAIN and CIPHER objects, and other device specific operations. All of these calls require no special privilege.

3.2.2 Support Programs

Several support programs have been created to provide management services for the PCED. The first, *Loadmi*, is a trusted program that facilitates loading the MI. Loadmi may only be invoked from within the Secure Xenix Trusted Shell, thereby providing a trusted path through which a System Administrator may reliably and safely load the MI. In addition to the MI, the PCED must also have a cryptographic key loaded in order to encrypt or decrypt data. *Loadkey* is the trusted support program that enables a user to load a key, again only via a trusted path. Before performing the driver call to load the key, Loadkey asks the user to enter the security level at which the key should be loaded. Loadkey then performs a safety check on the user by requiring that the level at which the key is to be loaded (as entered by the user via the trusted path) is equal to the current security level of the user process. In addition, Loadkey ensures that the desired security level of the key is within the PCED maximum and minimum security levels. Loadkey is then able to inform the driver of the current sensitivity level of both the CIPHER object, which is the security level at which the key will be loaded, and the PLAIN object which is the device minimum security level, by making the driver call to load the key. Both Loadmi and Loadkey are configured such that they can make the privilege system call to load the MI and key respectively.

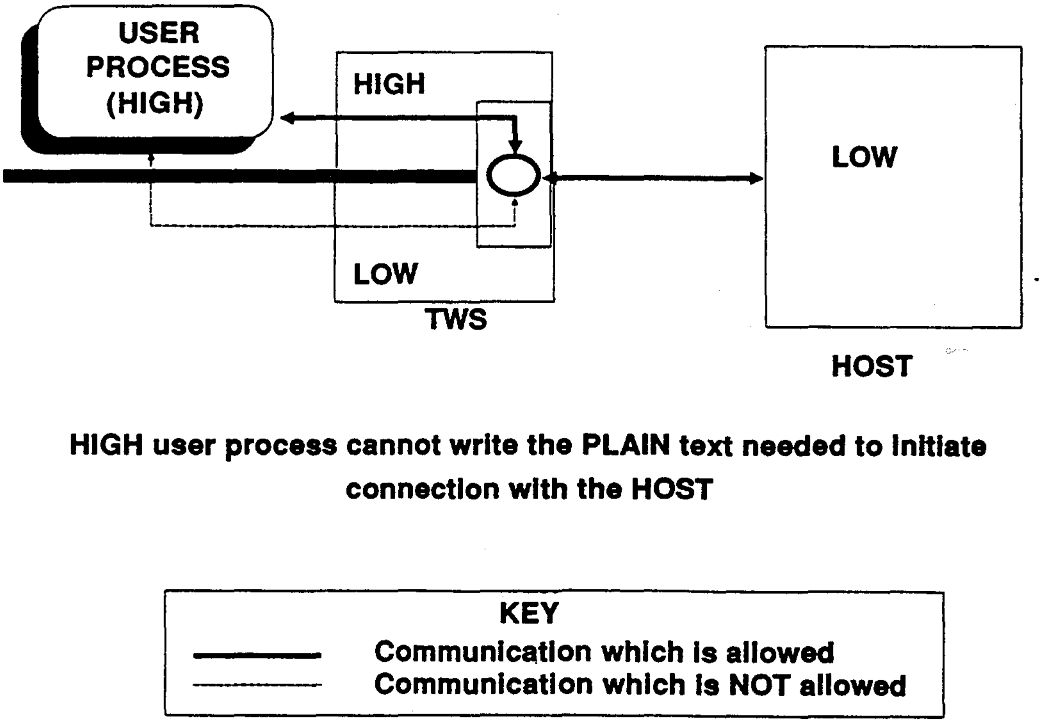
Once a key is loaded it can stay active throughout more than one usage of the PCED. Thus, one individual may load a key for use by several other individuals. Alternatively, users can have different keys which allow them to communicate with several different "communities of interest", in effect establishing multiple "crypto-nets". A user can always destroy a currently loaded key by hitting the reset button on the plain/cipher switch. The TWS procedures for handling and controlling cryptographic keys is completely compatible with current key management doctrine specified for the PCED system.

Showlabel, another trusted support program, may be called either from the Trusted Shell or from a user application. Its purpose is to retrieve the security level of the PLAIN and CIPHER objects from the PCED driver and display them to the user. If a key is not loaded at the time Showlabel is called, a message indicating so is displayed to the user. By executing this support program within the Trusted Shell, the user can determine the current security levels associated with the PCED in a reliable and unambiguous fashion.

3.2.3 A More Flexible Trusted Interface

In order to enhance the usability of the PCED, the program *Filter* was created to provide a more flexible trusted interface to the PCED for use by an application program. Specifically, the need for a trusted filter arose from the desire to support remote logins to untrusted network hosts for the purpose of file transfers and electronic mail. For example, consider the scenario where g_low is set to LOW, the key is loaded at HIGH, and the PCED RS-232 port is connected to a single-level host at LOW. In order to login to the host, a user would have to login to the TWS at LOW, which allows the user to send the appropriate commands to retrieve a file via the PLAIN object. However, since the user is logged in at LOW, any data that is decrypted by the PCED will be discarded by the driver because the user process is not permitted to read from the CIPHER object (see Figure 4). On the other hand, if the user were to login in at the security level of the key (i.e. HIGH) in order to read from the CIPHER object, the user process would not be able to write to the PLAIN object since the process is not able to "write down" to LOW. Therefore the user would be unable to send the appropriate plain text commands to the LOW level host to initiate the file retrieval process (see Figure 5).

Filter was designed to solve this dilemma. Filter logically resides between the PCED driver and a user application. In order to access the PCED via Filter, an application would interact with Filter much the same that it would with the PCED driver. However, Filter allows an application to specify a *log file* at the level of


USER
PROCESS
(HIGH)
HIGH
LOW
TWS
LOW
HOST
KEY
Communication which is allowed
Communication which is NOT allowed

HIGH user process cannot write the PLAIN text needed to initiate connection with the HOST

Figure 4
Unable to Establish Host Connection

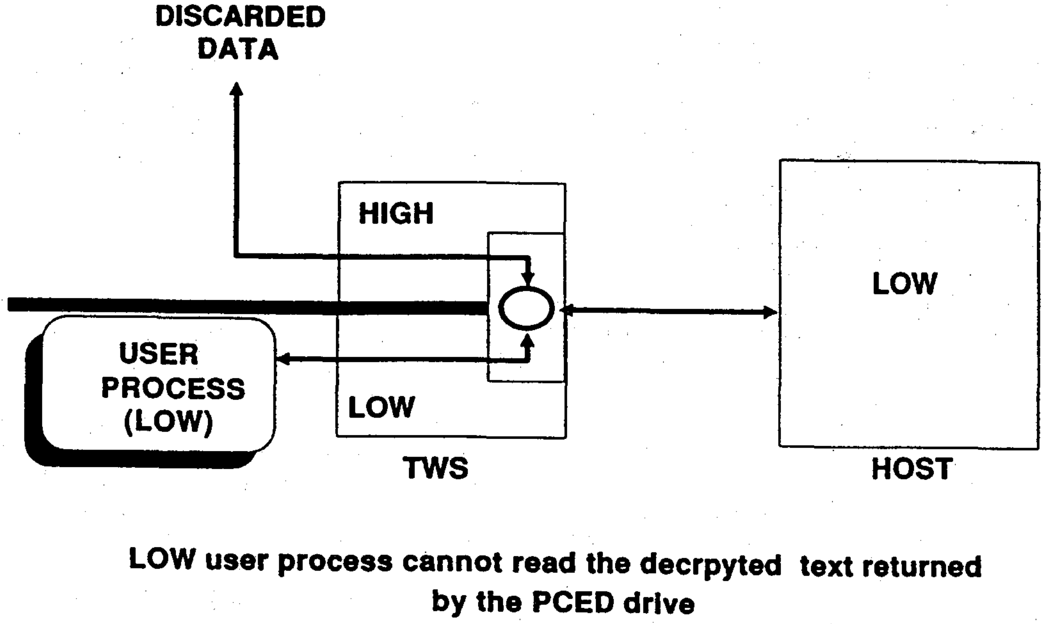

DISCARDED DATA
HIGH
LOW
USER PROCESS (LOW)
LOW
TWS
HOST
LOW user process cannot read the decrpyted text returned by the PCED drive

Figure 5
Cipher Data Lost

the key in which data from the CIPHER object can be saved for later retrieval by the user (see Figure 6). In order for Filter to provide this logging service, it is given sufficient privilege to read and write both the PLAIN and CIPHER objects simultaneously. Because Filter manages multiple levels of sensitive information it must duplicate the access mediation performed by the PCED driver.

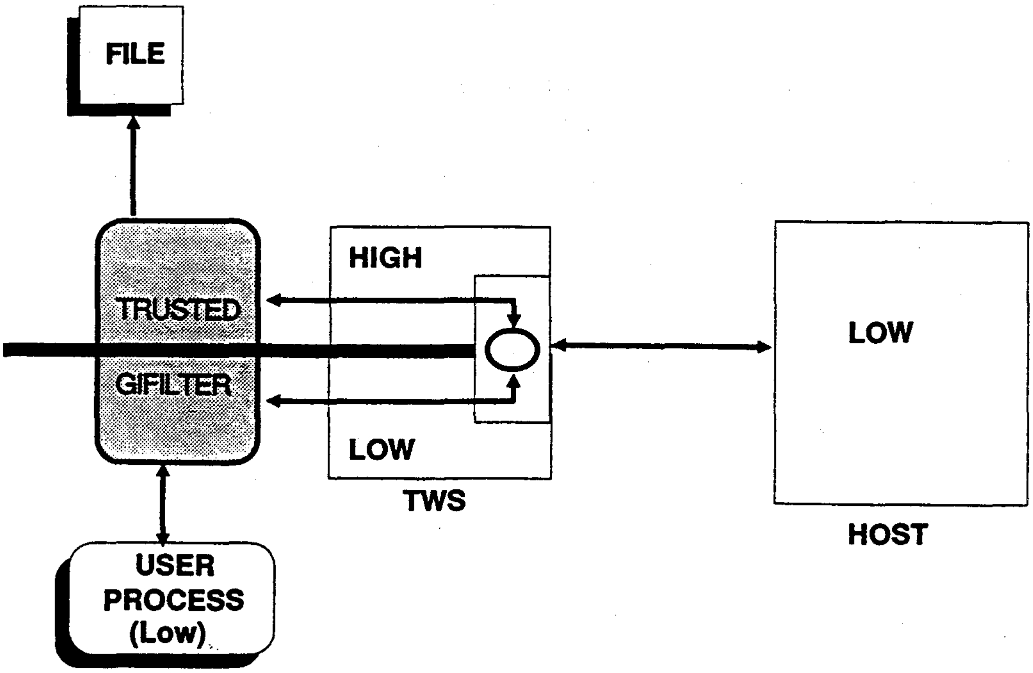
3.2.4 User Applications

Currently the TWS provides two untrusted user applications, *Comm* and *Term*. Both applications provide an interactive interface between the user and the Filter program, which then interfaces with the PCED driver. Through a series of menus and submenus, the user has the ability to send and receive data from other users as well as issue additional commands to the PCED. The data sent may originate either from a file or directly from the keyboard. Data received may be stored in a file, displayed on the screen or both. Users may also issue commands to change communication parameters (e.g., data rate, parity, stop bits, bits per character), toggle between the EMAIL and LINK mode of operation, change the screen or terminal configuration, (e.g., split screen, echo, carriage return insertion), or perform special Filter commands (e.g., sending a high level file, logging all PCED output to a high level file).

Comm provides an emulation of original DOS software provided with the PCED system. Comm is most effective for use when two TWS systems are directly communicating. Term is an enhanced user application which allows PCED equipped systems to indirectly communicate via intermediate systems (e.g., a centralized unclassified host, DDN). In effect, Term provides the user with a powerful terminal emulation in addition to the communication functions provided by Comm. In order for two PCED systems to communicate, they both must use the same key.

3.2.5 Future Directions

Comm and Term currently provide simple communications and file transfer capabilities (and in the case of Term, terminal emulation). We envision adding more user friendly communications features, such as Kermit compatible



Trusted Gifilter saves CIPHER text, which previously would have been discarded, in a file which is later accessable by the USER.

Figure 6
Problem Solved with Filter

applications, in the future. It also expected that more sophisticated applications, such as a mail facility, can be adapted to use to the PCED. Since all these applications can be developed entirely with untrusted code, no re-evaluation of the TWS system will be necessary for such future development.

Future versions of the TWS will use other cryptographic technology, such as those being developed via the commercial COMSEC Endorsement Program (CCEP), and employ more advanced forms of key management and off-line cryptography, as this technology becomes commercially available.

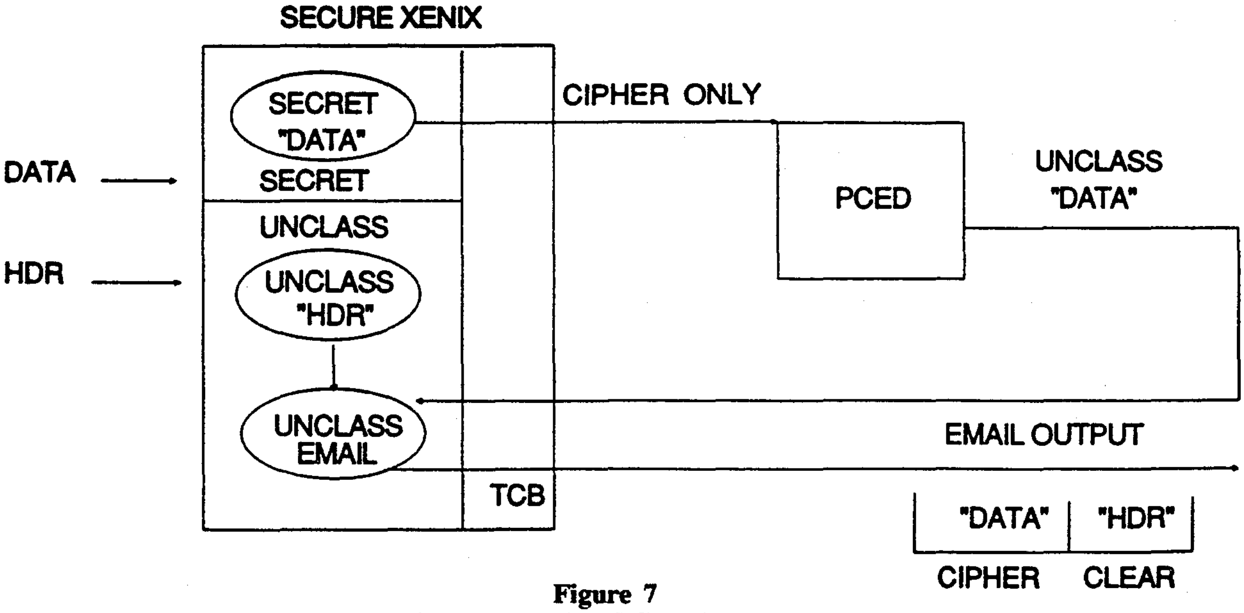


Figure 7
The TWS with Crypt using PCED in Loop Back Mode

We also hope to provide the ability to use the TWS for file encryption. Specifically, the PCED system is designed for encrypting information as it leaves a personal computer and decrypting information as it enters a personal computer. Comm and Term, used in conjunction with Filter is an example of using the PCED in an approved manner to perform end to end encryption. TIS has created another trusted application, *Crypt*, that has also been designed to perform end to end encryption by using the PCED in a "loop back" mode (see Figure 7). Crypt provides a mechanism by which encrypted text leaving the PCED communication port is fed back to the same system via a separate RS-232 communications port. The encrypted text is then appropriately labeled at a lower sensitivity level (which is commensurate with g_low), allowing the encrypted text to be processed using normal mail or file transfer applications written for Secure Xenix. On the receiving end, Crypt would send the encrypted data through the PCED for decryption. The resulting clear text is labeled at the higher security level of the original text (i.e., at the key level).

Using the PCED in loop back mode is technologically a safe and secure mode of operation as discussed in [11].

4. EVALUATION STRATEGY

Any credible integrated INFOSEC product must reaffirm the evaluation results of the COMSEC and trusted systems technology used in the product. From the onset of the TWS effort, the issues of re-evaluation for the TWS as a single integrated product have been a principal concern. In January 1988, after over a year of informal and formal interaction with representatives of several NSA INFOSEC organizations, TIS formally submitted a request for evaluation of the TWS as an INFOSEC product. Given that any single COMSEC or trusted system evaluation takes several years to complete, it was clear that a re-evaluation for the TWS must heavily rely upon past results in order for the re-evaluation effort to be practical and economical. Our strategy for the evaluation of the TWS is two-fold:

(1) Demonstrate that the modifications required for the TWS integration are all outside the "COMSEC boundary"; and

(2) Have the TWS re-evaluated by the NCSC as a trusted network component based upon a previously evaluated trusted operating system.

Since the TWS integration effort requires no modification to the PCED hardware or its operational characteristics, it appears that the first half of this strategy will be relatively simple to resolve. The remaining half of the evaluation strategy is the more difficult and is discussed in the remainder of this section.

4.1 The TWS as a Trusted Network Component

Secure Xenix is currently under evaluation as a B2 trusted operating system as defined by the TCSEC. As such, the product is technically only evaluated for use in a "stand-alone" mode. The TWS goal is to extend this evaluation to include distributed operation by having the TWS re-evaluation conducted under the *Trusted Network Interpretation* of the TCSEC [12], specifically as a network component as described in Appendix A of the TNI. This is a logical progression as the difference between Secure Xenix and the TWS is the inclusion of the PCED and supporting software whose intent is to securely interconnect more than one system.

While the evaluation of the TWS is referred to as a "re-evaluation", it is not the intent to evaluate the TWS (including the entire Secure Xenix operating system) from scratch. Rather, in order to make the entire evaluation effort economically feasible, the intent is to have the TWS evaluation "catch up" to the Secure Xenix evaluation such that the NCSC can apply its current knowledge from the on-going Secure Xenix evaluation to the TWS evaluation. Thus, we expect that the burden for the TWS evaluation is one of arguing that the modifications to Secure Xenix are both themselves secure and do not invalidate any of the previously evaluated Secure Xenix protection features. This type of evaluation is unique to both our and the NCSC experiences, and is somewhat analogous to the NCSC RAting Maintenance Program (RAMP) [13] (which currently addresses systems at or below B1).

While the system modifications are relatively simple, the evaluation effort is expected to be somewhat more involved. Every TNI requirement must be examined and argued for its completeness. This examination includes not only the functional aspects of the systems such as mandatory and discretionary access controls, but the assurance requirements such as formal modeling, descriptive top level specifications (DTLS), covert channel analysis, and configuration management. Each of these issues must be addressed through a combination of TWS additional, and Secure Xenix existing, features, documentation, and analysis results.

4.2 Product and Evaluation Status

Due to Trusted Information System's acquisition of Secure Xenix in June of 1989, the evaluation status for TWS has been radically modified. Prior to the acquisition, the NCSC had complete a preliminary technical assessment of the TWS product which culminated in a Preliminary Technical Report (PTR) written by NCSC evaluators. The PTR recommendation, which stated that the TWS should be accepted for the design analysis phase of the NCSC product evaluation program, was accepted by NCSC management. TIS and the NCSC both agreed that this evaluation would be economically feasible only if it did not require a complete re-evaluation of the underlying evaluated operating system. It was our intent to complete the evaluation of the TWS in conjunction with Secure Xenix.

At approximately the same time that NCSC management accepted the PTR recommendations, the future of Secure Xenix as an IBM product became unclear, effectively stopping all Secure Xenix *and* TWS evaluation activities. Ultimately, TIS's acquisition favorably resolved the evaluation and product status of Trusted Xenix (formerly Secure Xenix). However, since TIS is now responsible for both Trusted Xenix and TWS, the practicality of evaluating TWS as a product separate from Trusted Xenix has become uncertain.

At the time of this writing, the issue of a separate TWS evaluation versus an integrated Trusted Xenix/TWS evaluation remains open. Regardless of the manner in which the evaluation is eventually conducted, we fully expect to have the TWS trusted software evaluated as part of a larger trusted system product.

5. CONCLUSIONS

The Trusted Workstation demonstrates at least two important concepts. First is that INFOSEC products are possible and economical with today's technology and should be encouraged in the market place. Second is that INFOSEC products can be built from existing and approved COMSEC and trusted systems technology when prudently integrated. Ultimately, protection of classified information will be dependent upon INFOSEC products such as the TWS. TIS plans to continue to push the state-of-the-art in INFOSEC product integration and expects that, as newer and better technology becomes available, the lessons learned in the TWS effort can be applied to create better, safer, and more usable integrated INFOSEC products.

REFERENCES

[1] CSC-STD-003-85, "Computer Security Requirements," U.S. NCSC, 25 June 1985.

[2] CSC-STD-004-85, "Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements," U.S. NCSC, 25 June 1985.

[3] *DoD Trusted Computer Systems Evaluation Criteria*, DOD 5200.28-STD, National Computer Security Center, Ft. Meade, MD, December 1985.

[4] "Program Offering, IBM Secure XENIX, version 1.1," International Business Machines, Federal Systems Division, Gaithersburg, MD, 22 February 1988.

[5] Gligor, V.D.,et al,. "On the Design and the Implementation of Secure Xenix Workstations," *1986 IEEE Symposium on Security and Privacy*, Oakland, CA, April 1986.

[6] Luckenbaugh, G.L., et al., "Interpretation of the Bell-La Padula Model for Secure Xenix," *Ninth National Computer Security Conf.*, Gaithersburg, MD, September 1986.

[7] "Secure Xenix Product Evaluation Bulletin," Report No. CSC-PB-004-87, National Computer Security Center, Ft. Meade, MD, 18 August 1988.

[8] Bell, D.E., and L.J. La Padula, "Secure Computer System: Mathematical Foundations," MTR-2547, Vol. I, The Mitre Corp., Bedford, MA, March 1973.

[9] La Padula, L.J. and D.E. Bell, "Secure Computer System: A Mathematical Model," MTR-2547, Vol. II, The Mitre Corp., Bedford, MA, May 1973.

[10] Bell, D.E., "Secure Computer System: A Refinement of the Mathematical Model," MTR-2547, Vol. III, The Mitre Corp., Bedford, MA, December 1973.

[11] Walker, S.T., "Network Security: The Parts of the Sum," *1989 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1989.

[12] NCSC-TG-005, *Trusted Network Interpretation of the TCSEC*, Version 1, National Computer Security Center, Ft. Meade, MD, 31 July 1987.

[13] "Trusted Product Evaluations: A Guide for Vendors," NCSC-TG-002, Version 1, National Computer Security Center, Ft. Meade, MD, 1 March 1988.