NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY/
NATIONAL COMPUTER SECURITY CENTER

14TH NATIONAL COMPUTER SECURITY CONFERENCE

October 1-4, 1991
Omni Shoreham Hotel
Washington, D.C.

PROCEEDINGS
VOLUME I

Information Systems Security:
Requirements & Practices

Welcome!

The National Computer Security Center (NCSC) and the Computer Systems Laboratory (CSL) are pleased to welcome you to the Fourteenth 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: Requirements & Practices," 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 emphasizing the integration of information security solutions. These presentations will provide you with some thoughtful insights as well as innovative ideas in developing your own solutions. 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.

[illegible]
JAMES H. BURROWS
Director
Computer Systems Laboratory

[illegible]
PATRICK R. GALLAGHER, JR
Director
National Computer Security Center

Conference

Referees

Carole Jordan	*Defense Investigative Service*
Dr. Maria M. King	*The AEROSPACE Corporation*
Leslee LaFountain	*Department of Defense*
Steven LaFountain	*Department of Defense*
Paul A. Lambert	*Motorola GEG*
Dr. Carl Landwehr	*Naval Research Laboratory*
Robert Lau	*Department of Defense*
Dr. Theodore M.P. Lee	*Trusted Information Systems, Inc.*
Steven B. Lipner	*Digital Equipment Corporation*
Teresa Lunt	*SRI International*
Dr. William V. Maconachy	*National Security Agency*
Sally Meglathery	*ISSA*
Dr. Jonathan Millen	*The MITRE Corporation*
Warren Monroe	*Hughes Aircraft*
William H. Murray	*Deloitte & Touche*
Noel Nazario	*National Institute of Standards and Technology*
Ruth Nelson	*GTE*
Peter Neumann	*SRI International*
J.D. Nichols	*Independent Consultant*
Steven Padilla	*SPARTA*
Nick Pantiuk	*Grumann Data Systems*
Donn Parker	*SRI International*
Richard Pethia	*Carnegie Mellon University*
Dr. Charles Pfleeger	*Trusted Information Systems, Inc.*
Kenneth Rowe	*Department of Defense*
Professor Ravi Sandhu	*George Mason University*
Marvin Schaefer	*Trusted Information Systems, Inc.*
Dr. Roger R. Schell	*GEMINI*
Emilie J. Siarkiewicz	*Rome Air Defense Center*
Suzanne Smith	*Los Alamos National Laboatory*
Brian Snow	*Department of Defense*
Professor Eugene Spafford	*Purdue University*
Mario Tinto	*Department of Defense*
James Tippett	*Department of Defense*
Eugene Troy	*National Institute of Standards and Technology*
LTC. R. Vaughn, USA	*U.S. Naval Academy*
Grant Wagner	*Department of Defense*
Kenneth vanWyk	*Carnegie Mellon University*
Howard Weiss	*SPARTA*
Roy Wood	*Department of Defense*
Carol Worden	*State of Minnesota*

14th National Computer Security Conference

Table of Contents

x Authors Cross Index

Tutorials

1 From Tuples to Trusted Subjects to TDI: A Brief Tutorial on Trusted Database Management Systems
John R. Campbell, National Security Agency

13 Tutorial Series on Trusted Systems
Joel E. Sachs, Dr. William F. Wilson, Arca Systems, Inc.

PAPERS (refereed)

15 Accreditation Strategy for the Air Force Satellite Control Network (AFSCN)
Lt Col William Price, USAF, Air Force Space Command
Michael O'Neill, Frank White, CTA, Inc.

25 An Analysis of Application Specific Security Policies
Daniel F. Sterne, Martha Branstad, Trusted Information Systems, Inc.
Brian Hubbard, SPARTA, Inc.
Barbara Mayer, Atlantic Research Corporation
Dawn Wolcott, MITRE Corporation

37 Another Factor in Determining Security Requirements for Trusted Computer Applications
David Ferraiolo, National Institute of Standards and Technology
Karen Ferraiolo, Grumman Data Systems

45 Apparent Differences Between the U.S. TCSEC and the European ITSEC
Dr. Martha Branstad, Dr. Charles Pfleeger,
Trusted Information Systems, Inc.
Dr. David Brewer, Gamma Secure Systems, Ltd.
Mr. Christian Jahl, Mr. Helmut Kurth, IAGB Software Technology

59 Auditing of Distributed Systems
D. Banning, G. Ellingwood, C. Franklin, C. Muckenhirn, D. Price,
SPARTA, Inc.

69 Building a Multi-Level Application on an Untrusted DBMS in a UNIX System V/MLS Environment - A Project's Experience
David S. Crawford, Canadian Department of National Defence

78 Building a Multi-Level Secure TCP/IP
Deborah A. Futcher, Brian K. Yasaki, The Wollongong Group
Ron L. Sharp, AT&T Bell Laboratories

88 The Cascade Problem: Graph Theory Can Help
John A. Fitch, III, Lance J. Hoffman, George Washington University

101 A Case Study for the Approach to Developing a Multilevel Secure Command and Control Information System
James Obal, Supreme Allied Commander Atlantic
William Grogan, Contel Federal Systems

110 Contractors and Computer Security--Awareness, Education, and Performance
Ronald E. Brunner, Ronald G. Brunner & Asssociates

120 Covert Channel Analysis Planning for Large Systems
Lee Badger, Trusted Information Systems, Inc.

137 Dealing With a Malicious Logic Threat: A Proposed Air Force Approach
Howard L. Johnson, Information Intelligence Sciences
Chuck Arvin, Earl Jenkinson, CTA, Inc.
Captain Bob Pierce, USAF, Electronic Security Command

147 Developing Applications on LOCK
Richard O'Brien, Clyde Rogers, SCTC

157 The Development of a Low-To-High Guard
Michelle J. Gosselin, MITRE Corporation

167 DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype
Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, University of California, Davis
Stephen E. Smaha, Steven R. Snapp, Haystack Laboratories, Inc.
James Brentano, Pacific Gas and Electric Company
Lt. Tim Grance, USAF, Daniel M. Teal, USAF, United States Air Force Cryptologic Support Center
Douglass L. Mansur, Lawrence Livermore National Laboratories

177 A Distributed Implementation of the Transform Model
Ravi S. Sandhu, Gurpreet S. Suri, George Mason University

188 Employee Privacy and Intrusion Detection Systems: Monitoring on the Job
Lorrayne J. Schaefer, The MITRE Corporation

195 Experience of Commercial Security Evaluation
Peter Fagan, Julian Straw, Secure Information Systems Limited

205 Experiences in Multi-Level Security on Distributed Architectures
Karl A. Siil, AT&T Bell Laboratories

215 An Expert System Application for Network Intrusion Detection
Kathleen A. Jackson, David H. DuBois, Cathy A. Stallings, Los Alamos National Laboratory

226 Formal Verification of a Network Security Device: A Case Study
Hicham N. Adra, William Sandberg-Maitland, CGI Information Systems & Management Consultants

237 A Framework for Advancing Integrity Standardization
Terry Mayfield, Stephen R. Welke, John M. Boone, Catherine W. McDonald, Institute for Defense Analyses

246 A Framework for Developing Accreditable MLS AISs
R. K. Bauer, J. Sachs, M. L. Weidner, W. F. Wilson, Arca Systems, Inc.

257 Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy
Marshall D. Abrams, Jody Heaney, Osborne King, Leonard LaPadula, Manette Lazear, Ingrid Olson, The MITRE Corporation

267 Honest Databases That Can Keep Secrets
Ravi Sandhu, Sushil Jajodia, George Mason University

283 Identifying and Controlling Undesirable Program Behaviors
Maria M. King

295 Improvement of Data Processing Security by Means of Fault Tolerance
Gilles Trouessin, Yves Deswarte, Jean-Charles Fabre, LAAS-CNRS & INRIA
Brian Randell, Computing Laboratory, The University Newcastle upon Tyne

305 Information Security: Can Ethics Make a Difference
Corey D. Schou, John A. Kilpatrick, Idaho State University

313 Information Security Risk Analysis and Risk Management: Which Approach?
Professor J.H.P. Eloff, K.P. Badenhorst, Rand Afrikaans University

328 Information Systems Security: A Comprehensive Model
Capt. John R. McCumber, USAF, Joint Staff, the Pentagon

338 Integrating B2 Security into a UNIX System
Kevin Brady, UNIX System Laboratories, Inc.

347 Knowledge Based Computer Security Advisor
William Hunteman, M. B. Squire, Los Alamos National Laboratory

357 The Logistics of Distributing a Smart Token
Dawn Brown, Department of Defense

362 A Method to Detect Intrusive Activity in a Networked Environment
L. Todd Heberlein, Biswanath Mukherjee, Karl Levitt, University of California

372 Model Based Intrusion Detection
Thomas D. Garvey, Teresa F. Lunt, SRI International

386 Notification: A Practical Security Problem in Distributed Systems
Vijay Varadharajan, Hewlett-Packard Laboratories

397 Output Perturbation Techniques for the Security of Statistical Databases
Kasinath C. Vemulapalli, Elizabeth A. Unger, Kansas State University

407 An Overview of Informix-Online/Secure
Rammohan Varadarajan, Informix Software, Inc.

417 Peeling the Viral Onion
Russell Davis, Planning Research Corporation, Inc.

427 Practical Models for Threat/Risk Analysis
Mark W.L. Dennison, Kalman C. Toth
CGI Information Systems & Management Consultants, Inc.

436 Predicate Differences and the Analysis of Dependencies in Formal Specifications
D. Richard Kuhn, National Institute of Standards and Technology

446 Preventing Weak Password Choices
Eugene H. Spafford, Purdue University

456 Putting Policy Commonalities to Work
D. Elliott Bell, Trusted Information Systems, Inc.

472 Reconciling CMW Requirements with Those of X11 Applications
Glenn Faden, Sun Microsystems, Inc.

480 Restating the Foundations of Information Security
Donn Parker, SRI International

494 The Role Of Network Security In A Methodology For Information Security Design And Implementation
Professor J.H.P. Eloff, Mr. A.J. Nel, Rand Afrikaans University

505 A Secure European System for Applications in a Multi-vendor Environment (The SESAME Project)
T. A. Parker, ICL Secure Systems

514 A Secure Quorum Protocol
Masaaki Mizuno, Mitchell L. Nielsen, Kansas State University

524 Security Guidance for VAX/VMS Systems
Debra L. Banning, SPARTA, Inc.

533 Sneakernet: Getting a Grip on the World's Largest Network
Captain James B. Hiller, USAF, Space and Warning Systems Center

543 A Socio-Technical Analysis of a USA National Computer Security Conference
Stewart Kowalski, Stockholm University & Royal Institute of Technology

533 Standardized Certification
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center

563 A Strategic Framework For Information Security Management
Rolf Moulton, BP America
Santosh Misra, Cleveland State University

572 A System Security Engineering Process
J. D. Weiss, AT&T Bell Laboratories

582 Teaching Computer Systems Security in an Undergraduate Computer Science Curriculum
Alfred W. Arsenault, Captain Gregory B. White, USAF,
U. S. Air Force Academy

598 Toward Certification, A Survey of Three Methodologies
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center

608 Trusted Distributed Computing: Using Untrusted Network Software
E. John Sebes, Richard J. Feiertag, Trusted Information Systems

619 Trusting X: Issues in Building Trusted X Window Systems or What's not Trusted About X?
Jeremy Epstein, TRW Systems Division
Jeffrey Picciotto, MITRE Corporation

630 Using Existing Management Processes to Effectively Meet the Security Plan Requirement of the Computer Security Act: The IRS Experience
Richard A. Stone, Joseph Scherer, Internal Revenue Service

634 Viruses in an OS/2 Environment: Remembrances of Things Past and a Harbinger of Things to Come
Kevin P. Haney, National Institutes of Health

644 Why Does Trusted Computing Cost So Much?
Susan Heath, Phillip Swanson, Daniel Gambel, Grumman Data Systems

PANEL Executive Summaries (unrefereed)

654 **PANEL:** Acquiring Computer Security Services and Integrating Computer Security and ADP Procurement
Dennis Gilbert, National Institute of Science and Technology
Barbara Guttman, National Institute of Science and Technology

655 **PANEL:** Compartmented Mode Workstation(CMW) Program Overview
Steven Schanzer, Moderator, Defense Intelligence Agency

658 **PANEL:** The Computer Emergency Response Team System (CERT System)
E. Eugene Schultz, Lawrence Livermore Laboratory
Richard Pethia, Software Engineering Institute, Carnegie Mellon University

663 **PANEL:** Computer Security Management and Planning
Christopher Bythewood, National Computer Security Center

664 **PANEL:** Cracking the Cracker Problem
Dorothy E. Denning, Moderator, Georgetown University
665 The Role of Technology
Matt Bishop, Dartmouth College

666 **PANEL:** Electronic Dissemination of Computer Security Information Executive Summary
Marianne Swanson, National Institute of Science and Technology
667 What Can Dockmaster Offer You?
Cindy Hash, Department of Defense

Session: Guidelines & Evaluations
669 Towards Mutual Recognition of Security Evaluations
Andrea Arnold, Digital Equipment Corp
Cornelia Persy, SIEMENS
Gottfried Sedlak, IBM

674 **PANEL:** Fielding COTS Multilevel Security Solutions: The Next Step
James Litchko, Trusted Information Systems Inc.

675 **PANEL:** Inference and Aggregation in Multilevel Databases: Research Directions
Teresa F. Lunt, Moderator, SRI International
676 Detecting and Evaluating Inference Channels
Thomas D. Garvey, SRI International
679 Inference Prevention in Databases: Data Design vs. Query Processing
Catherine Meadows, Naval Research Laboratory
680 Challenges in Addressing Inference and Aggregation
LTC. Gary Smith, USA, National Defense University
681 Approaches to Handling the Inference Problem
Bhavani Thuraisingham, The MITRE Corporation

684 **PANEL:** Military and Telecommunications Security: Specialized Methods
Richard Lefkon, Moderator, New York University
685 Malicious Code Prevention for Embedded Computer Weapon Systems
Debra L Banning, Gail M. Ellingwood, SPARTA
689 Computer Viruses as Electronic Warfare
Myron Cramer, Booz-Allen & Hamilton
690 Preventing Virus Insertion Through Switches
Ed Fulford, Northern Telecon
693 Nuclear Disaster and The Millennium Horse
Richard Lefkon, New York University

Session: National Issues
695 Reduced Defense Spending Increases the Need for Trusted Systems
Carole S. Jordan, Defense Investigative Service

696 **PANEL:** 1991: A Year of Progress in Trusted Database Systems
John R. Campbell, Moderator, National Security Agency
698 Recent Developments in Some Trusted Database Management Systems
Bhavani Thuraisingham, The MITRE Corporation
701 Oracle and Security: Year in Review 1990-91
Linda L. Vetter, Oracle Secure Systems
704 1991 SYBASE Secure Products: Executive Summary
Helena B. Winkler-Parenty, SYBASE

706 **PANEL:** Requirements and Experiences
Dennis Gilbert, National Institute of Science and Technology

708 **PANEL:** Risk Management
Irene Gilbert, National Institute of Science and Technology

709 **PANEL:** Specifying, Procuring, and Accrediting MLS System Solutions
Joel E. Sachs, Arca Systems, Inc.

714 **PANEL:** Trusted Applications in the Real World
Stephen Walker, Trusted Information Systems Inc.

715 **PANEL:** Winning Strategies in Information Systems Security Education, Training, and Awareness
W. V. Maconachy, Moderator, Department of Defense

FROM TUPLES TO TRUSTED SUBJECTS TO TDI: A BRIEF TUTORIAL ON TRUSTED DATABASE MANAGEMENT SYSTEMS

John R. Campbell
National Security Agency
9800 Savage Road
Fort George G. Meade, Maryland 20755-6000
301-859-4387

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 being addressed. Trusted products are being introduced commercially. Vendors and potential vendors of trusted products include Atlantic Research Corporation (ARC), DEC, Informix, Infosystems Technology, Ingres, Oracle, Sybase and Teradata. A second significant gain in 1991 is the completion of the Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria (TDI). The TDI extends the evaluation classes of the Trusted Computer System Evaluation Criteria (TCSEC) to trusted applications in general, and to database management systems in particular. The evaluation of trusted database systems has been started by the National Computer Security Center. As of this writing, two products were under evaluation; others are in preparation for evaluation.

Database security is maturing somewhat as a discipline. Some very tough issues are being examined and understood. For example, we know a lot more about the causes of, the problems associated with and the potential solutions for polyinstantiation now than when we put it in a contract to force people to look at the problem. There has been good research and development in this area. For example, Rome Labs is sponsoring the development of a B2 system, Oracle is examining the relationship between integrity and confidentiality and we are supporting the development of a trusted database system with A1 Mandatory Access Control. Research is being done, among other things,on distributed, multimedia, and object-oriented trusted database systems.

This tutorial gives the background, describes the issues and offers some proposed solutions for database security. The title was deliberately chosen. The "tuple" is a record instance or row in a table. I will briefly discuss database systems, and, more specifically, relational database systems, as systems based on this model are currently the most widely used systems. "TCB- Subsets" is a key concept in database security because, as an application, it sits on other software, perhaps an operating system. The concept permit efficient evaluation of trusted software in a very high skill, labor intensive process. The "TDI" is an important work, not only because it aids evaluators of trusted database systems but because it deals with layering and applications in general.

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 management 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.

Client-server architectures are becoming popular. Data could be stored in a database on a larger computer or server. The data is then usually accessed by smaller computers called clients. Many users on personal computers or workstations could then efficiently access a large database on a larger server. The clients and servers are connected by perhaps a LAN. A problem is that the system: clients, server and LAN must recognize and protect security labels. This recognition may not be easy, especially if each component comes from a different vendor.

Finally, distributed database systems have added additional complexities to the security problem. The data in these system 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? What form of concurrency do you use? We are seeing repeatedly that data integrity conflicts with confidentiality. How do you get both? What are the tradeoffs? 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. While this is still a research topic, object-oriented systems also appear to be easier to secure.

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. If you can modify or destroy my data or otherwise deny me access to my data, then the data is not secure. Consequently, our definition agrees with what the Strategic Defense Initiative calls "security *" which includes nondisclosure, data integrity and availability.

Our definition also includes ease-of-use as a requirement for "security". If the user or security administrator finds a system too difficult to use because of security, then the security features will not be used. This is easily done as most security features on database systems are optional. A goal then is to build systems that appear to be very similar to vanilla systems, that use standards such as the Structured Query Language (SQL), and that are compatible as possible to previous databases and database systems.

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?

Concurrency controls are integrity controls that enable many users to run their programs and access the database at the same time. They prevent incorrect interactions between transactions. In this way throughput and availability of the database management system are enhanced. Standard controls however, can be used as signalling channels, thereby harming nondisclosure. This area is a research topic and work is being done.

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 client-server and distributed database work.

Some are handled as database security issues. The problems of inference and aggregation are not unique to database systems. They deal with relationships between data. However, the inference and aggregation problems are exacerbated by database management systems, because these systems are designed to easily manipulate large quantities of data. Some issues, such as granularity, are unique to database security.

Some issues are treated as database security issues because they had to be solved before a trusted database system could be built. Layering and TCB subsets were studied for trusted database systems but they apply to trusted applications in general.

Finally, there are issues that seem to be unique to the distributed DBMS. How do you update replicated data or recover in a secure fashion? These also are research questions.

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 TDI will add insight into the evaluation of database management systems and other applications.

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. Good software engineering would also dictate layering.

A TCB subset M is a set of software and/or firmware and/or hardware that mediates the access of a set S of subjects to a set O of objects on the basis of a stated access control policy P and satisfies the properties:

1. M mediates every access to objects in O by subjects in S;
2. M is tamper resistant; and
3. M is small enough to be subject to analysis and tests, the completeness of which can be assured. [NCSC91]

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. Again, 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

There, fortunately, has been much activity in implementing commercial versions of trusted database systems. The vendors include ARC, Informix, Oracle, Sybase and Teradata. Other trusted systems are being developed.

The most popular implementation is a Trusted Computing Base (TCB) implementation where the DBMS enforces Mandatory Access Control on the DBMS objects. Part of the DBMS is a trusted subject. Performance here is independent of tne number security levels and compartments. Evaluation is more complex and difficult. Sybase and Informix are examples.

In another Trusted Computing Base implementation, the operating system provides the mandatory access control, while both operating system and DBMS may provide discretionary access control. The evaluation should be easier. However, each combination of security level and compartment requires a separate database instance. Performance should decrease with increasing numbers of security level/compartment combinations. Unclassified data may be separately stored as such. Oracle's product offers the choice of either this or the first approach.

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.

This method should require minimal additional trusted code and minimal changes to an existing DBMS, and therefore be less costly to build. Because the DBMS is untrusted, there may be covert channel problems [LAND88] and more direct attacks. ARC is an example.

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.

One additional approach has been called the "distributed" approach. Here, one untrusted computer is used for each security level/compartment combination. A central trusted computer handles computer selection and query parsing. Two varieties exist. In the first, each machine has security combination. In the second, each machine has all the data up to that security combination. In the first, joins must be done in the central computer; in the second, joins can be done in the untrusted computers. Both could require much hardware. We know of no vendor examples. Research is being done.

NATIONAL COMPUTER SECURITY CENTER (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

These are typical factors that would be considered in a trusted implementation.

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 Oracle and Ingres.

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 and some B-level operating systems are here. An evaluation aid, the Trusted Database Interpretations,has been published. Trusted DBMS products are being produced. 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]

domain - The set of objects that a subject has the ability to access. [NCSC91]

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]

subset-domain - A set of system domains. For evaluation by parts, each candidate TCB subset must occupy a distinct subset domain such that modify-access to a domain within a TCB subset's subset-domain is permitted only to that TCB subset and (possibly) to more primitive subsets. [NCSC91]

trusted subject - A subject that is permitted to have simultaneous view and alter access to objects of more than one sensitivity level. [NCSC91]

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 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.
[NCSC91]	National Computer Security Center, Trusted Database Management Criteria, 1991.
[WINK89]	Winkler-Parenty, C. E., "Can You Trust Your DBMS", Database Programming & Design, July 1989, pp. 50-59.

Tutorial Series On Trusted Systems

Joel E. Sachs and Dr. William F. Wilson
Arca Systems, Inc.
2841 Junction Ave., Suite 201
San Jose, CA 95134
408-434-6633

Schedule

Tuesday, October 1	**0900**	**Trust Fundamentals - Part I**
	1030	**BREAK**
	1100	**Trust Fundamentals - Part II** **Network Security Fundamentals**
	1200	**LUNCH**
	1400	**System Solutions and Security** **Distributed Security**
	1530	**BREAK**
	1600	**Certification & Accreditation** **Trusted Integration**
	1730	**ADJOURN**

Description

These tutorials are based on Arca Systems' public and on-site Information Security Courses and experience learned in applying Arca's security consulting and engineering services to systems solutions. Arca provides support to its clients on both secure MLS system solutions and security products in all facets of trusted system design, analysis, development, implementation, testing, verification, integration, certification and accreditation. Arca has focused particularly on both trusted applications development and trusted integration of many products into secure system solutions. The tutorials relate experience from supporting systems integrators, applications developers, and end-users, as well as product vendors, who are addressing security in a variety of MLS system solutions for command and control, communications, and intelligence systems, development environments, and embedded systems.

The tutorials will be presented in lecture format with questions and answer periods. While there is a logical flow between the tutorials, each tutorial will be presented as a separate unit so that conference attendees can attend any or all of them. The morning tutorials concentrate on information security basics and the afternoon ones focus on addressing security in system solutions. The tutorials are intended to introduce many and varied security topics as opposed to exploring them in-depth. Brief descriptions of each tutorial identified above follows:

Trusted Fundamentals - Part I focuses on security and (TCSEC) trust concepts. Topics include security policies, mandatory and discretionary access controls, identification and authentication; security mechanisms, reference monitors, trusted computing bases, trusted path, least privilege; and assurance, formal and informal verification, covert channel analysis, security design analysis, security and penetration testing.

Trusted Fundamentals - Part II focuses on the TCSEC Evaluation Classes. The tutorial presents an overview of the TCSEC, its evaluation classes, and the NCSC evaluation process.

Network Security Fundamentals focuses on basic points in network security and gives an overview of the TNI. Topics include network security concerns and services, the structure of the TNI and its Evaluation Classes for both network TCBs and network components, and an overview of the TNI evaluation process.

System Solutions and Security focuses on system-wide security requirements in the context of system solutions. Topics include system solution characteristics, models, and development methodologies; and system-wide security problems, concerns, and threats and vulnerabilities.

Distributed Security focuses on the role of network security in today's distributed system solutions. Topics include system composition and interconnection, single system views versus interconnected automated information systems [AISs], cascading, encryption, and trusted network interfaces.

Certification & Accreditation focuses on the development of the certification evidence and inputs and decision process for accreditation. Topics include an overview of the certification and accreditation process, critical considerations, modes of operation, risk analysis, overall assurance requirements, and collecting system-wide evidence and assurance.

Trusted Integration focuses on integration issues that arise when developing and integrating secure and MLS system solutions. Topics include system-wide views of security policy, mechanism, and assurance; system, subsystem and component level interpretations for the roles of security policies, security policy models, and security top level specifications; security impact on the development methodology; and overview of security trade-offs.

ACCREDITATION STRATEGY FOR THE AIR FORCE SATELLITE CONTROL NETWORK (AFSCN)

By Lt Col William R. Price
Air Force Space Command/LKXS, Peterson AFB, CO 80914
Michael E. O'Neill, Ph.D. and Frank O. H. White
CTA Incorporated
7150 Campus Drive, Colorado Springs CO 80920

ABSTRACT

This paper examines the accreditation approach for a large, complex computer network, namely the Air Force Satellite Control Network. The network represents many existing computer networks, and as such, the approach for accreditation has broad application to the computer security community. The paper provides a brief background and history of the AFSCN. The accreditation approach is then described, followed by specific implementation stages for accreditation. The last section addresses "lessons learned" in the development of an accreditation strategy for the complex network.

Section 1-INTRODUCTION

This paper describes an ongoing effort to accredit a large, military communications-computer network. Although the paper describes a particular network, the Air Force Satellite Control Network, the authors believe it is representative of many existing networks and the approach taken has broad application to the computer security community. Functionally, the network supports the tracking, telemetry and commanding of military satellites. Telemetry from satellites provides status and health functions of on-orbit platforms (e.g., navigation, orientation, status of power system). The network typically does not process data collected or transmitted by satellite mission sensors (e.g., weather data). It provides both voice and data connectivity among satellite control sites throughout the world. This "real world," operational network has evolved over many years without the benefit of modern computer or network security theory and practice. Accreditation of the network is challenging from both a technical and management perspective.

The large and complex AFSCN has evolved over the last three decades. It employs a variety a variety of technologies. These technologies range from second generation computers and patch panel based communication systems to modern computers, workstations and computer controlled communication switching systems using fiber optics. Although security was not ignored in the design and evolution of the network, most AFSCN security protections predate the Trusted Computer Security Evaluation Criteria (TCSEC) and its Trusted Network Interpretation (TNI). A significant part of the accreditation effort is to devise or "reverse engineer" the overall network security concept and and document it.

Several organizations are involved in the management, operation and use of the AFSCN. Air Force Space Command (AFSPACECOM), an Air Force major command, is the network manager and, consequently, the Network Designated Approving Authority (DAA). Other organizations involved are Air Force major commands, DoD activities and civilian agencies such as the National Aeronautics and Space Administration (NASA). No one organization has complete control over the network, and the accreditation of the network must involve mutual benefit and agreement rather than the dictates of a single organization.

The remainder of this paper describes the ongoing efforts to accredit the network. Section 2 describes the AFSCN in more detail, providing information on its basic functions as well as the complexity encountered in addressing security architecture and security management relationships in the AFSCN community. Section 3 discusses the approach to accreditation that is being pursued and the considerations that determined the approach. Section 4 describes the accomplishments to date and remaining activities planned. Section 5 describes our lessons learned which may of value to security managers involved in approving other communications-computer networks.

Section 2-BACKGROUND

AFSCN Components and Functions

The AFSCN provides spacecraft owner/operators (Air Force, NASA and others) the capability to track their satellites, send them commands and downlink health and status telemetry. These tracking, telemetry and commanding (TT&C) functions are depicted in Figure 1, AFSCN Concept of Operations. Several key AFSCN facilities, also referred to as AFSCN components in this paper, are shown in Figure 1. These components are briefly described below:

- Mission Control Centers (MCCs) are owner/operator facilities that remotely monitor and control spacecraft from launch to the end of their on-orbit life. MCCs maintain tracking information on their satellites and contact them as required to send commands and download telemetry related to spacecraft health and status. MCCs are operated by a variety of military and civilian agencies (AFSPACECOM, Air Force Systems Command (AFSC), NASA and others). Some MCCs support specialized Research and Development (R&D) spacecraft while others manage mature operational satellite systems. Most Air Force MCCs are located at the Consolidated Space Test Center, Onizuka Air Force Base (AFB), California or the Consolidated Space Operations Center, Falcon AFB, Colorado. Some MCCs supporting joint NASA/DoD operations are located at Johnson Space Center near Houston, Texas.

- There are nine Remote Tracking Stations (RTSs) located worldwide that provide spacecraft interface to the AFSCN environment. Most RTSs have two or more independent antennas and associated ground equipment sets for acquiring, tracking and communicating with spacecraft. A wide variety of radio frequency links and data link protocols can be supported by the ground equipment. RTS equipment, especially that installed by the Automated Remote Tracking Station (ARTS) program, can be remotely controlled by an MCC interfaced through the AFSCN communications network.

- AFSCN has redundant network control nodes at Onizuka and Falcon AFBs. Each node consists of a Resource Control Complex (RCC) and a Communications Control Complex (CCC). The RCC schedules network resources and directs configuration of network facilities to support the unique requirements of each spacecraft contact support mission. Under direction of its associated RCC, the CCC establishes connectivity called for in the contact mission support schedule. The CCC establishes circuits for commands, status and control, timing, telemetry and secure voice. Circuits to perform these functions are set up on both primary wideband and narrowband communications links to the RTSs. Bandwidth and data formats vary greatly from mission to mission due to the characteristics of the supported satellite.

- There are many other facilities in the AFSCN environment that support those described above. Software development facilities, test laboratories, satellite and RTS simulators, test driver systems and command centers are just examples. A variety of development, operations and logistics organizations operate these and a host of contractor support systems.

The AFSCN provides the communications services for satellite operators in MCCs to contact and control their spacecraft. The following scenario describes a typical satellite contact support mission:

Air Force Satellite Control Network (AFSCN) Concept Of Operations

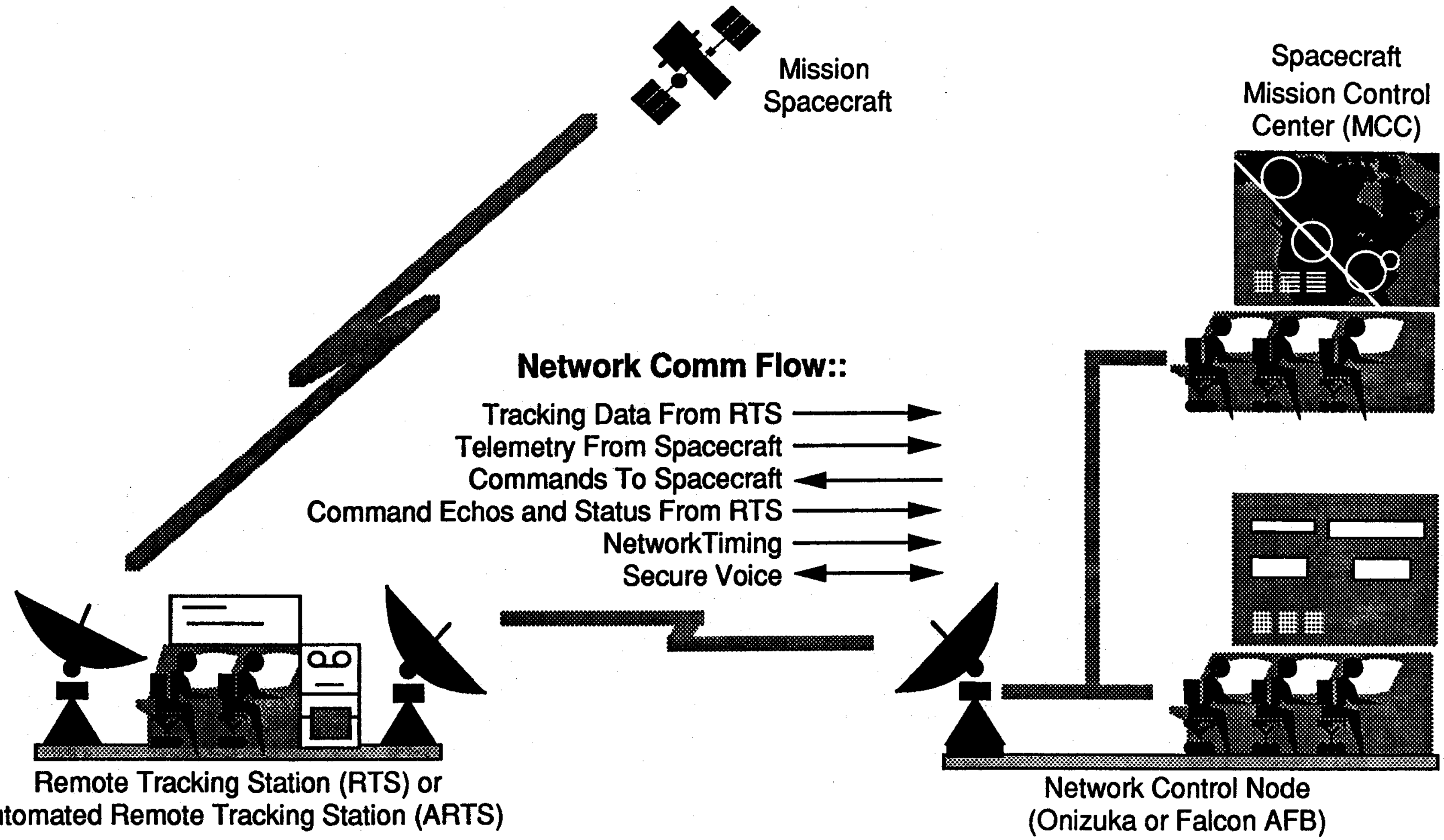

Mission Spacecraft
Spacecraft Mission Control Center (MCC)
Network Comm Flow::
Tracking Data From RTS →
Telemetry From Spacecraft →
Commands To Spacecraft ←
Command Echos and Status From RTS →
NetworkTiming →
Secure Voice ↔
Remote Tracking Station (RTS) or Automated Remote Tracking Station (ARTS)
Network Control Node (Onizuka or Falcon AFB)

Figure 1 AFSCN Concept Of Operations

- Usually weeks in advance, an MCC coordinates with the AFSCN's primary RCC at Onizuka to schedule a contact support mission for its spacecraft. The contact support mission calls for the MCC to be connected to an RTS that has the satellite in its line-of-sight long enough for the required TT&C operations.

- The RCC schedules the network resources (the RTS, the mission-unique circuit mix on primary and alternate communications paths, and CCC interface to the MCC). This configuration is called a contact support mission string. The string may be needed for a few minutes or several hours depending on the spacecraft, its orbit and the TT&C functions to be performed. Hundreds of contact support missions are requested each week by MCCs representing many different spacecraft programs. There are sufficient network resources to support several simultaneous contact support missions (i.e., multiple mission strings operating in overlapping time frames), but often not all requests can be accommodated at the same time. The RCC continually deconflicts these competing requirements, often negotiating alternate times and network configurations with MCCs. The RCC manually generates a seven-day projection and a final 24-hour schedule to task network resources.

- Just prior to mission time, the RCC coordinates with the RTS and CCC via secure voice to configure the mission string. Establishing the mission string involves a combination of automated processes and manual patching by operators in the RCC, CCC, MCC and RTS. After verifying circuit connection on both primary and alternate communications systems, as well as proper functioning of RTS equipment, the RCC transfers computer control of the mission string to the supported MCC.

- After the RTS acquires the spacecraft, the MCC's Contact Support Processor receives and records tracking data from the RTS and sends commands to both the spacecraft and RTS equipment to start telemetry transfer. Typically, commands to the spacecraft and downlinked telemetry are protected at the Secret level. This end-to-end communications security is provided by peer encryption devices on the spacecraft and on the front end of the MCC processor. Unclassified mission data (status and control messages, timing, etc.) exchanged between the MCC and RTS are protected at the Unclassified Sensitive level. This transmission security protection is provided by bulk encryption devices on network communications links.

- When the MCC completes its spacecraft contact, the RCC disconnects the MCC and resumes control of network resources. Equipment and circuits in the mission string are returned to the pool of network resources for allocation to other scheduled missions. After disconnect, the MCC processes the telemetry data and often transfers it to support facilities for further reduction and analysis.

Some History

Until recently, AFSCN facilities were developed, owned and operated primarily by AFSC, an Air Force major command responsible for research and development. This changed in 1987 when the newly formed AFSPACECOM began to assume operational responsibility for AFSCN assets not dedicated to research programs. Over time, AFSPACECOM became owner/operator of the RTSs around the world, AFSCN satellite and terrestrial transmission systems and RCCs and CCCs at Onizuka and Falcon. They also activated some new Air Force MCCs at Falcon. AFSC retained responsibility for R&D spacecraft and continues to support them from MCCs at the Onizuka. With the transfer of most AFSCN operational systems to AFSPACECOM, the Colorado Springs based command was designated the overall AFSCN Manager and assumed primary responsibility for security management.

The Network Security Environment

Two AFSPACECOM organizations were tasked to implement security management: (1), the Headquarters DAA who is responsible for approving operation of all computer and communications systems operated by AFSPACECOM; (2) the AFSCN Security Manager in the 2 Space Wing (2 SWG) who is responsible for day-to-day management of the AFSCN system security program. In assessing the state of security management in AFSCN, the DAA and Security Manager found the following:

- On the whole, competent security programs and security engineering had been put in place by various program offices over the years, but these typically focused on the computer system or facility being fielded or modified at the time. There were numerous security accreditations on file for individual computer systems and even a few for logical groupings of computer and communications facilities. The various accreditations for individual systems were for different modes of operation and security classification levels. Several security classification guidance changes were under consideration, but lacking an overall security concept or policy, assessing the impact of these proposed changes on the network was virtually impossible. Like most complex networks in place before promulgation of the national network security policy, the existing system and facility accreditations were like pieces of a complex puzzle. No one had yet begun to assemble the puzzle.

- There were many ideas how the puzzle should be assembled, but none of these seemed practical from a security standpoint. Reviews of planning and program management documents, as well as extensive interviews with managers of key development, operations and support organizations, revealed multiple network definitions. Each of these definitions made sense when seen through the eyes of their advocates, but no one definition provided a useful basis for understanding security-relevant services, facilities, interfaces and bounds. Although differing in detail, most definitions seemed all encompassing, driving the security analyst to examine unfathomable detail: scores of computer facilities and systems, hundreds of interfaces and a labyrinth of connectivity. In short, there was not a well articulated security architecture and there was not enough time or money to perform a network security analysis using the complex definitions offered up by various constituents in the AFSCN community.

- The network environment was highly dynamic, with literally hundreds of hardware and software upgrades underway at any one time. These upgrades were advocated and managed by a large number of organizations and programs, often competing for resources. Configuration control across the network was extremely complex. Network security enjoyed a very low priority in all this.

- Security management roles of the various commands, agencies and organizations involved in the community needed to be more clearly defined. There were many competent and highly motivated security managers throughout the community, but they focused on the facilities and systems within their sphere of influence. They were aware of evolving national guidance on network security, but the concepts and mechanisms of network security had yet to be institutionalized.

- There were differences in interpretation of Air Force computer security policy among the organizations. There was a perception that AFSCN systems would have to be scrutinized by both the individual System DAA and the Network DAA. Some organizations resisted such an approach where systems that had received security approval (System DAA) would have to be reviewed and approved again by an outside organization (Network DAA). These organizations felt that the Network DAA had no authority or responsibility for their operations. Another concern was that the individual System DAAs would duplicate efforts and, moreover, could reach different conclusions based on differing interpretations of Air Force computer security policy.

Section 3-THE APPROACH

This section outlines the approach and methodology that was used to develop an accreditation strategy for the network. The challenge was to develop means and methods that would build consensus and approval for the approach and subsequent development of the AFSCN Network Security Policy.

Simplification Vs. Complexity

To get a reasonable handle on the network from a security perspective, a workable network definition was needed. Stepping back from the complex network definitions available in the community, we developed a simple model for trying to understand the security relevant relationships among the various facilities and systems in the AFSCN environment. This model, depicted at Figure 2, allocates the various facilities and systems to one of three layers. The model consists of the "Core Network" surrounded by two additional layers, External Interfaces and Support Components.

The Core Network consists of all the tracking, transmission, switching, and resource control facilities required to connect a spacecraft to its respective MCC, whether that MCC is operated by AFSPACECOM, AFSC, NASA or some other activity. The Core Network is anchored by the control nodes at Onizuka and Falcon AFBs. The other components in the Core Network are RTSs and ARTSs.

The External Interface layer of the model contains all AFSCN facilities that connect to the Core Network for TT&C services. In the main, these are MCCs, but other AFSCN components do connect to the Core Network from time to time for the purpose of TT&C testing and training. Also included in the External Interface layer are two satellite control networks dedicated to specific programs: Defense Metrological Support Program and the Global Positioning System.

All other AFSCN components are allocated to the Support Component layer of the model. These components frequently connect to External Interfaces for data analysis, software maintenance and other functions, but they rarely, if ever, connect to the Core Network. Allocation of components to this layer, where they have no direct impact on Core Network operational or security services, greatly simplified the complexity in our accreditation task.

We decided to look at the AFSCN as analogous to a telephone company providing service to its customers. As the sole operator of the Core Network, AFSPACECOM provides spacecraft owner/operators TT&C services much in the same way a telephone company provides telecommunications services to its customers. By thinking of the AFSCN and the Core Network in this manner, it allowed us to develop a strategy for accreditation that could be supported by the myriad of owners and users of AFSCN assets.

The telephone company view provided both technical and management benefits. Technically, the view allows a "divide and conquer" approach. The Core Network is the communications subsystem and provides for communication of unclassified data. It mediates MCC accesses to RTSs and knows nothing about individual users (people) in the MCCs. From the perspective of an MCC, the MCC communicates with a peripheral (i.e., the spacecraft) through the Core Network. The management benefit of this view is that it divides the network along organizational lines. The components of the Core Network are the responsibility of AFSPACECOM, the Network DAA. Individual MCCs and other External Interfaces are the responsibility of their operating commands, primarily AFSPACECOM and AFSC. The approach

Layered Approach to AFSCN Security

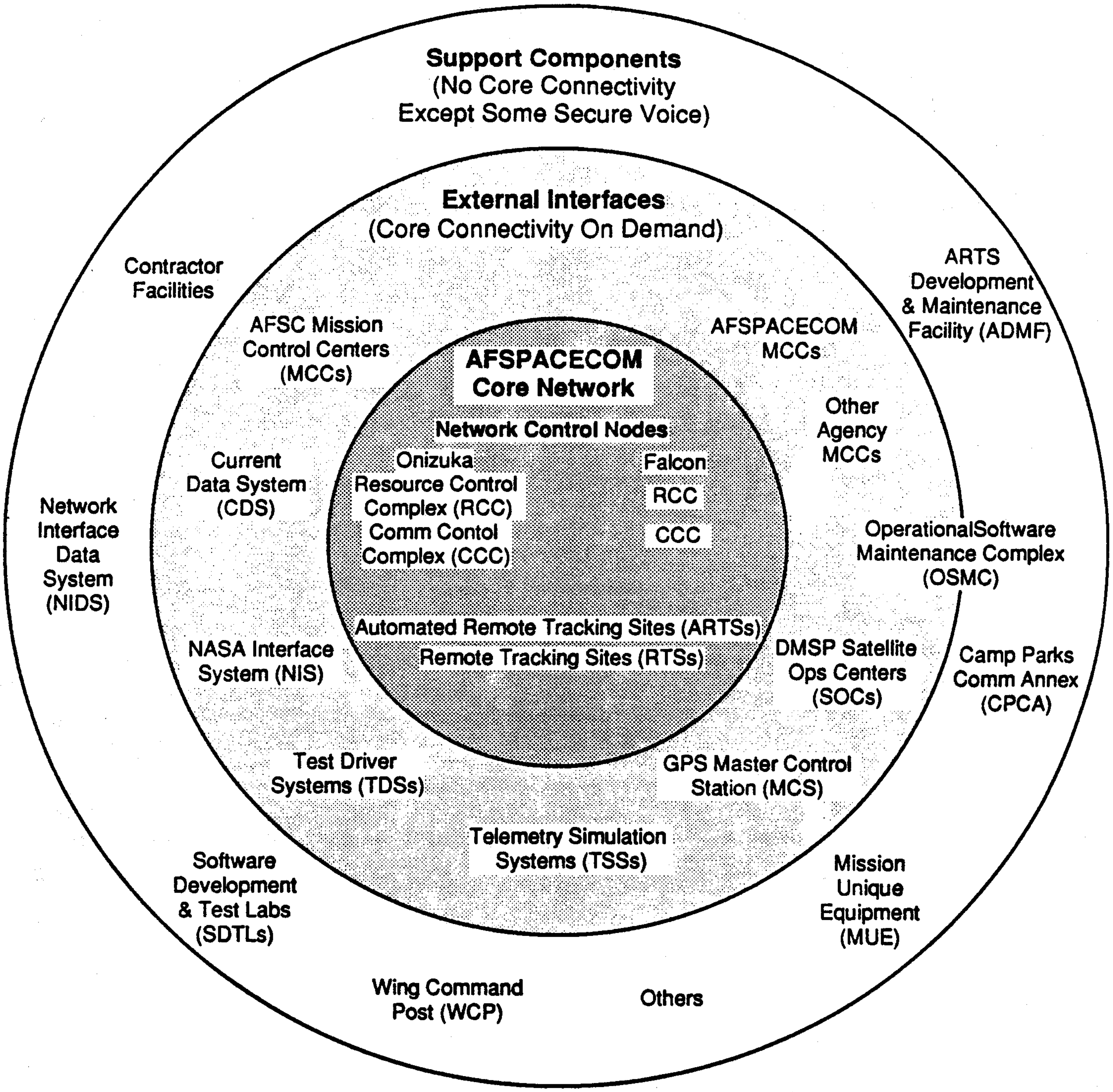


Figure 2
Layered Approach to
AFSCN Security

to network approval involves mutual actions of the DAAs with the mutual benefit of secure operations. The Network DAA certifies to the MCC DAA the security properties of the Core Network (assuming the External Interface satisfies certain conditions of connection). Correspondingly, the DAA for the MCC certifies compliance with network connection rules and approves operation of the MCC (assuming the Core Network maintains its security properties). This concept of mutual security assurance applies to all External Interfaces, not just MCCs.

Consensus Building

The development of a consensus among key AFSCN community players was absolutely necessary for a successful accreditation strategy. The newly formed AFSCN Security Working Group was instrumental in this effort.. This group represents the breadth of the AFSCN community, including security professionals, space operations personnel, developers and support managers. As a forum for presenting and refining the security model, it played a key role in getting support for the security model, the security concept of operations and their codification in the AFSCN Network Security Policy.

The Network Security Policy

The Network Security Policy first defines the AFSCN in terms of the layered model discussed earlier: a Core Network (tracking stations and communications that transport real time data and voice services in support of spacecraft contact missions), External Interfaces to the Core Network, and Support Components. This layered approach provides a method to understand who has what authority, who is responsible for what components and who is held accountable for what AFSCN security matters.

The policy defines three major security objectives for the Core Network. They are: 1. Network Confidentiality (non-disclosure), 2. Network Integrity and 3. Network Assurance of Service. An integrated program of protective security measures is employed across the Core Network for each security objective. Responsibility for implementing Core Network security objectives and protections are discussed as are specific connection rules for External Interfaces.

Dissecting the network into understandable components provided the framework to identify management responsibilities, authority and a means to provide accountability for the security policy objectives. Basically, the policy calls for command/agency DAAs to accredit individual AFSCN components in accordance with AFR 205-16 or equivalent agency security policy directives. They certify to the Network DAA that these formal accreditations are accomplished and that their components are compliant with all applicable requirements of the Network Security Policy.

Operational Perspective

In developing the Network Security Policy with the AFSCN community, the Network DAA and representatives from organizations operating network components, recognized the need for simple, streamlined security procedures that minimize impact on network operations. Based on the concept of mutual trust and security competence among DAAs, the Network Security Policy established the Letter Of Assurance (LOA) as the administrative mechanism for the Network DAA to maintain an ongoing assessment of the network security posture. The LOA is a one page document whereby DAAs for network components (Core Network and External Interfaces) certify to the Network DAA that their components are compliant with the security protection standards and connection rules in the Network Security Policy. The LOAs provide the basis for the Network DAA to accredit the Core Network and authorize connection of its External Interfaces.

Section 4-IMPLEMENTATION ACTIONS

Having developed a general consensus within the AFSCN security community regarding the Network Security Policy, detailed implementation of the policy is underway. Some of the major steps are discussed below:

- The first step is to create a network security management structure for implementing and enforcing the Network Security Policy. This structure includes all organizations that operate components in the Core Network and in the External Interface layer of network security model. With the Network Security Policy laying out the authorities, responsibilities and key relationships, this structure is headed by the Network Security Manager (NSM). The NSM is responsible for overall implementation and enforcement of the policy across the network and across organizational lines. The System Security Working Group, with representatives from all component organizations, is the NSM's advisory group for surfacing and resolving policy issues. Network Security Officers (NSOs) appointed in each Core and External Interface component execute the NSM's program on a day to day basis. These "hands-on" security managers implement and enforce detailed security procedures, investigate incidents and implement corrective actions. The security management structure also includes the Network DAA and DAAs from the various organizations that operate Core and External Interface components.

- The NSM must develop, coordinate and publish a Network Security Plan that provides detailed guidance for managing the Network Security Program. This document must include methods and standards governing risk analyses and security test and evaluations.

- Procedures must be developed and implemented to enforce the Network Security Policy in the requirements and configuration control processes. The DAAs and NSM must have visibility of new requirements and network changes so that they may assess security impacts and favorably influence implementation.

- All Core Network components and External Interfaces must be accredited by their respective DAAs. Most computer systems and facilities have current Interim or Final approvals to operate; however, some communications and tracking facilities in the Core Network have never been accredited. Additionally, some existing accreditations are nearing three-years of age and must be reaccomplished.

- As DAAs accredit components, they will certify these approvals to operate to the Network DAA through the Network Security Manager. As discussed earlier, the Letter Of Assurance will be the vehicle for this certification.

- When all the Core Network components and External Interfaces are accredited, the Network DAA will be able to accredit the network.

Section 5-LESSONS LEARNED

The AFSCN had been in place for many years before promulgation of national network policy. As such, it represented an evolving collection of complex components. It was a significant challenge to take this amalgam of components and develop a strategy for its accreditation. The lessons learned in this process are as follows:

- Develop a forum of security professionals for consensus building and negotiation of critical security considerations in the network. Look for people who have a vested interest in development of the forum and ultimate accreditation of the network.

- Define a simple, understandable architectual security model. It was necessary to develop a single but realistic definition of the AFSCN from a security standpoint. The concept of the Core Network was developed in order to bound the network. This was done through consensus building and negotiation with the security professionals from throughout the AFSCN community.

- Develop a spirit of mutual trust among the developers and operators that represent various organizational interests. This involves divorcing the Network DAA from the detailed risk analysis activities and holding the various organizations accountable for their portion of network risk assessment and accreditation. The Network DAA should, however, assure network integrity to its users and operators by initiating a Network Security Policy that precisely defines security protection mechanisms and connection rules. The basis for assurance from the Network DAA and DAAs accrediting network components and interfaces are Letters of Assurance that certify compliance with the Network Security Policy.

- The Network DAA should have a realistic and flexible attitude toward the network. It would be unrealistic to think that the Network DAA could shut down the AFSCN.

- Focus on a security management structure for implementing and enforcing the Network Security Policy. When defining network security responsibilities and authority, look for an existing organizational structure that can fulfill these duties whenever possible. For example, Network Security Officer responsibilities can be assigned to personnel who currently perform Computer System Security Officer functions at the various network facilities.

- Get senior management involvement from all organizations at critical stages of the accreditation process. Continually brief senior management on progress and strategy. This will develop the necessary support when critical decisions are required that cut across organizational boundaries.

AN ANALYSIS OF APPLICATION SPECIFIC SECURITY POLICIES

Daniel F. Sterne Martha A. Branstad Brian S. Hubbard[†] Barbara A. Mayer[‡] Dawn M. Wolcott[§]

Trusted Information Systems, Inc., Glenwood, Maryland

Abstract

The TCSEC [20] is concerned primarily with the DoD confidentiality. As a result, for many applications, systems that satisfy the TCSEC may nevertheless provide an insufficient base of security policy enforcement. This paper summarises a study whose objective is the identification of a broader range of security policies that merit automated support, particularly in tactical computer systems.

The study analysed operational requirements of a collection of tactical and non-tactical application scenarios. Synopses of several example scenarios are presented, and the findings of the study are discussed. The study suggests that while many policies are application specific, there exists a core of policy elements common to a broad range of such policies, and that this core merits automated support in future trusted systems.

Keywords: *security policy, access control, roles, integrity, denial of service.*

1 Introduction

The TCSEC [20] is oriented primarily toward confidentiality policies, and in particular, the protection of classified information from disclosure to insufficiently cleared individuals. As a result, systems that satisfy the TCSEC may fail to address other important security requirements, particularly those associated with tactical military applications. If systems capable of satisfying broader ranges of security requirements are to be constructed, the security policies that underlie these requirements must be more clearly articulated. To the extent that these policies may be application specific, it is important that policy elements common among them be identified, that these elements become candidates for automated support in future trusted systems.

This paper summarises the initial phase of a project whose ultimate objective is the construction of a prototype system that can be configured to support a range of application specific security policies, and in particular, policies associated with military systems [24]. The objective of this initial phase is the identification of security policies and common policy elements that merit automated support in tactical computer systems.

The U.S. Department of Defense (DoD) is under increasing pressure to use commercial-off-the-shelf (COTS) hardware and software, and to avoid procuring customised system components. Because commercial systems,

This work was funded by DARPA through RADC contract F30602-89-C-0125

[†]Currently with SPARTA, Inc., Columbia, MD

[‡]Currently with Atlantic Research Corp., Hanover, MD

[§]Currently with the MITRE Corp., McLean, VA like tactical systems, may also need to support policies not addressed directly by the TCSEC, another objective of the initial phase is to examine the commonality of commercial and other non-tactical policies with those of the tactical realm. If commonality exists, systems designed to support commercial and non-tactical security policies may be able to support tactical policies as well.

1.1 Approach

This study could have proceeded based purely on conjecture and an abstract conceptual view of tactical operations. That approach seemed overly speculative, and unlikely to produce meaningful results. To keep the study more closely tied to reality, a somewhat different approach was taken. A sampling of applications "scenarios" was selected, and for each scenario, information was gathered and then analyzed. The tactical scenarios chosen deal with the Navy's Aegis combat system, the command, control, and communications interactions associated with the Air Force nuclear weapon release process, and Army field operations and support services. The non-tactical scenarios concern government procurement document preparation and release, commercial accounting and data processing, air traffic control, and medical information system usage.

If the security policies associated with these scenarios were clearly understood and had been clearly articulated by their associated organizations, it would have been sufficient for this study to have collected and catalogued existing policy statements; little if any policy analysis would have been required. However, the distinction between a "security policy" and other kinds of regulations, operational procedures, and critical system requirements has not been clearly established. Consequently, for many organizations, it is not clear that distinct security policy statements actually exist, apart from those concerned with confidentiality.

In the absence of such policy statements, the study proceeded by examining operational and system requirements for each scenario. For scenarios dealing with existing organizations and systems, to the extent practical, these requirements were collected from technical articles and discussions with knowledgeable individuals. For scenarios dealing with future systems whose requirements and impacts on organizations have not yet been completely established (e.g., CALS [9]), incomplete information about operational requirements was augmented by educated guesses.

Each scenario was then analyzed to identify underlying security policies and policy characteristics. While the analysis produced results the authors believe are useful, these results are of necessity partly subjective; policy statements cannot be mathematically derived from operational requirements, but can only be loosely inferred. Moreover, the analysis was not exhaustive; it did not attempt to consider all requirements or identify all possibly relevant security concerns. The analysis of each scenario concentrated on a small set of security concerns that seemed most fundamental with respect to the overall mission and threats. Consequently, the results reported here are not intended as a definitive analysis. Rather, they represent an illustrative sampling of security policies in which an emphasis has been placed on security concerns other than confidentiality.

1.2 Organization

This paper is organized as follows. First, a few fundamental definitions are given. Next, excerpts from the security analysis of three example scenarios are presented to illustrate the range of security policy elements identified in tactical and non-tactical scenarios. The examples are followed by a summary and discussion of the study's findings, and a short section on future work.

2 What Is A "Security Policy" ?

The scenarios examined in the study encompass a wide spectrum of critical operational and system requirements. Given the objective of identifying security policies, in particular, policies beyond confidentiality, it became apparent early on that a means for distinguishing security policies from other kinds of critical requirements was needed. Since recent trends in terminological usage have tended to blur the distinction between criticality and security, a set of fundamental definitions was developed for use in the study. These definitions, described below, also distinguish between policies that govern human activity and those that govern automated processes on a computing system. Furthermore, these definitions describe a somewhat different view of security than that implied by the maxim "confidentiality, integrity, and assured service" [27, 7, 22]. A more complete discussion of these definitions can be found in [25].

2.1 Definitions

Security Policy Objective - A statement of intent to protect an identified resource from unauthorised use. The statement must identify the kinds of uses that are regulated. A security policy objective is meaningful to an organisation only if the organisation owns or controls the resource to be protected.

This definition establishes the primary notion of security upon which the other definitions are based: protection of tangible assets from unauthorised use. Examples of security policy objectives include protecting classified information from unauthorised disclosure or modification, preventing unauthorised distribution of financial assets, preventing unauthorised use of long-distance telephone circuits, preventing unauthorised dispensing of prescription drugs. The notion of a security policy used here is broader than that of the TCSEC, which is concerned with protecting a single kind of resource: information.

Organisational Security Policy (OSP) - The set of laws, rules, and practices that regulate how an organisation manages, protects, and distributes resources to achieve specified *security policy objectives*. These laws, rules and practices must identify criteria for according individuals authority, and may specify conditions under which individuals are permitted to exercise or delegate their authority. To be meaningful, these laws, rules, and practices must provide individuals reasonable ability to determine whether their actions violate or comply with the policy.

An OSP describes how a security policy objective is to be manifested in the routine activities of the organisation. The OSP definition is patterned after the security policy definition given in the TCSEC glossary,[1] but addresses protection of resources other than information. In addition, it explicitly cites the authorization of individuals as fundamental to the notion of a security policy, and allows authorisation to be based on attributes other than clearance and need to know. For example, authorisation may be based on job title, employer, training, licensing, enrollment, or membership.

Automated Security Policy (ASP) - The set of restrictions and properties that specify how a computing system prevents information and computing resources from being used to violate an *organizational security policy.*

An ASP specifies what a trusted system is trusted to do. The ASP for a TCSEC-oriented trusted system (class B or higher) typically includes the Bell-LaPadula properties [3], labeling requirements for human readable output, I&A-oriented restrictions (e.g., minimum password length), audit capture requirements, and so forth.

[1] "The set of laws, rules, and practices that regulate how an organisation manages, protects, and distributes sensitive information."

2.2 The Meaning of "Security Policy" In this Paper

This study is concerned with *organizational* security policies, that is, laws, rules, and practices that govern the activities of people. The analysis of organisational security policies is a prerequisite for analysis of automated security policies. Throughout this paper, the terms "security policy", and "policy" are used for brevity, but are intended to refer to organisational rather than automated security policies.

3 Roles

In the sections that follow, the term "role" [2, 15, 26] occurs frequently. We will use the term role to mean a named group of rights; these rights are permissions to access, operate on, or otherwise use resources in particular ways. A financial officer role might include rights to disburse financial assets (by signing checks) and to approve release of corporate financial information. The role of payroll clerk may include the right to examine employee salary data. The role of pharmacist includes the right to dispense drugs but not prescribe them; that right belongs to the physician role. A DoD security officer role might include rights to add new user accounts to a classified computing system and to control the system's audit data collection. Individuals belonging to an organisation are assigned to roles and are then able to exercise the rights associated with those roles. Consequently, roles are a means of naming and describing many-to-many relationships between individuals and rights.

Role exclusion rules may be associated with roles. These rules place constraints on the ability of individuals to be authorised for roles or to assume roles for which they are otherwise authorised. For some roles, there may be a limitation on how many individuals can be concurrently active in the role [15]. For example, in certain military organisations, only a single individual may be able to assume the role of watch officer at a time. Other individuals who are otherwise authorised to assume the watch officer role, cannot assume the role until it has been relinquished. Some combinations of roles may be considered "conflicting" because together they provide more authority than the organisation permits any one individual to hold; there may be a prohibition against any one individual being assigned (authorised for) more than one of these. For example, in a commercial corporation, an individual may be prohibited from acting as both a financial officer and a financial auditor. This kind of exclusion rule is equivalent to so-called "static" separation of duty, as defined in [19], and discussed elsewhere in the literature [6, 18].

4 The Aegis Combat System

The Aegis combat system is a sophisticated shipboard combat system used in U.S. Navy cruisers and destroyers [8]. The Aegis system includes a variety of sensors, including radar and sonar, and weapons, including surface-to-air missiles, surface-to-surface missiles, miscellaneous anti-submarine devices, guns, and small multi-purpose helicopters. These assets are monitored and controlled from the Combat Information Center (CIC), a room containing numerous operator consoles and large screens used to display situation maps and tactical summaries. In order to support mission requirements for high fire power and rapid response to threats, the Aegis system provides extensive automated response capabilities that can be programmed by the ship's crew.

Three organisational security policies were identified from descriptions of the Aegis system. These concern the prevention of 1) unauthorised disclosure of classified information, 2) unauthorised modification of information, and 3) unauthorised release of weapons. Each is discussed in a subsection below.

4.1 Information Disclosure Policy

The information disclosure policy is based on well-established DoD regulations and is directed at protecting classified information from individuals lacking sufficient clearances. Crew members may be uncleared, cleared for shipboard tactical information, or cleared for both intelligence and shipboard tactical information. In addition, uncleared visitors may occasionally be aboard. Most members of the crew are cleared for tactical information, which includes targeting data, locations of friendly forces, mission plans and situation tactics, and information about capabilities and limitations of sensors, weapons, and other equipment. A small fraction of the crew may, in addition, be cleared for access to intelligence information.

Both kinds of information are protected by physical and procedural security measures. Armed guards prevent unauthorised individuals from boarding the ship when it is in port. While at sea, access to the CIC and to the intelligence room is controlled by locks on entry doors. When information is transmitted among the ships in the fleet, communications security measures are employed to prevent eavesdropping.

4.2 Information Modification Policy

The information modification policy is concerned with preventing unauthorised individuals from supplying, changing, or deleting intelligence and tactical information. To a lesser extent, it may also be concerned with preventing authorised individuals from modifying such information in an clearly erroneous manner. This policy is not explicitly articulated, but has been inferred by the authors from descriptions of operational procedures.

Intelligence information and tactical information must only be accepted from designated sources. Designated sources may be organisations, or individuals assigned to particular job functions. Designated sources vary according to the type of information. Accepting information from sensors, computers, or other equipment is authorised if the equipment is operated under the auspices of a designated source organisation or individual. The authority to act as a designated source for a particular kind of information constitutes a role.

Cleared shipboard personnel are authorised to extract, derive, delete, enter, or otherwise modify tactical information. Similarly, personnel with intelligence clearances are authorised to modify intelligence information. When authorised individuals make such modifications, they are expected to employ any applicable designated processing methods or algorithms[2] so that modifications are minimally subjected to simple error checks. In some cases, however, the organisation must rely primarily on the considered tactical judgment of senior officers to ensure that information modifications are valid, i.e., consistent with reality and the intentions of superiors. Moreover, all authorised individuals are trusted not to introduce intentional inaccuracies into protected information, except as required for sanitisation purposes.

Ships in the fleet may share tactical data (e.g., concerning potential targets) in digital form via radio-based ship-to-ship communications. As a result, console operators on one ship have a limited measure of authority to influence (modify) another ship's tactical information base. The extent of this authority is constrained by protocols and algorithms that are used to resolve conflicts among multiple information sources. Depending on the circumstances and kind of information involved, conflicting information received from other ships may replace or be added to the information generated by a ship's own sensors and crew. Alternatively, conflicting information may be discarded, or mathematically combined.[3] Thus for each ship, an authorisation distinction is made between console operators on that ship, and those on other ships in the fleet; these constitute different roles. Except for cleared members of fleet crews and designated information sources, no other individuals have authority to modify shipboard information.

[2] These may be embedded in the ship's computer programs.
[3] Planned for future system upgrades.

4.3 Weapon Release Policy

The weapon release policy is directed at preventing weapons, especially missiles, from being released without appropriate authorisation. Only the ship's Commanding Officer (CO) has the authority to order the release of weapons, although he may delegate this authority to the Tactical Action Officer (TAO). Although several individuals on a ship may be authorised to assume the TAO role, only one can assume it at a time; this is an example of a role exclusion rule. The CO or TAO can order (authorise) one or more of the combat system console operators to release a weapon. A weapon release order can be given directly to the console operator, or may propagate downward to the operator through the chain of command. Similarly, only the CO and TAO have the authority to order the creation, modification, enabling or disabling of programmable automated weapon release rules called "doctrine statements". The Combat System Coordinator (CSC) is the only role given authority to enter or alter these statements and typically is authorised to do so only when specifically directed. Furthermore, typical operating procedures dictate that doctrine statements be written on paper and signed by the CO prior to being entered into the system by the CSC.

4.4 Policy Summary

The security policy objectives for this scenario include preventing unauthorised disclosure and modification of information, and preventing unauthorised release of weapons. Authorisation to use these resources is contingent on clearances, roles and role exclusion rules, delegation of authority, and non-repudiatable (signed) orders.

5 Nuclear Command, Control, and Communications

The principal requirements of the nuclear command, control, and communications (NC3) system are 1) rapid response to authorised orders directing the release of nuclear weapons, 2) prevention of unauthorised weapon release, and 3) prevention of unauthorised disclosure of classified information associated with deployment plans and the release process.

5.1 Weapon Release Policy

A nuclear weapon release requires collaborative actions on the part of multiple individuals, each of whom has been assigned one of three specific roles. The civilian authority authorises the use of nuclear weapons. The military authority generates specific targeting orders that must comply with previously established plans. These orders are then carried out by launch control officers. This division of authority amongst the civilian authority, the military authority, and the launch control officers constitutes separation of duty. No unilateral action by any individual in any of these roles, by itself, should allow a nuclear weapon release to be successfully initiated. Forced collaboration among these roles during the release process is accomplished via cryptographic procedures. In addition, a split knowledge policy among the individuals assigned the role of military authority requires that at least two of these individuals collaborate (by combining secrets) before they are able to execute a release successfully. Stringent source authentication requirements play a central role in the protocols used by these roles during their interactions; in some cases, the protocol prohibits a role from proceeding with its duties without having successfully authenticated the source of a received directive.

Following authorisation, two-person or N-person controls are used extensively; each launch control officer is assigned to a team, and is prohibited from carrying out launch control related activities unless authorised and accompanied by his team member(s). These controls prohibit a single launch control officer from accessing launch control information, facilities, authenticators, and cryptographic materials.

5.2 Denial of Service Policy

The denial of service policy for the NC3 system is directed at preventing unauthorised individuals from inhibiting an authorised release of nuclear weapons. (Having such a policy does not preclude the possibility that some individuals may be authorised to prevent weapon releases, for example, after cessation of hostilities.) This policy is manifested in a host of personnel, physical, and communications security measures that are beyond the scope of this discussion. We note, however, that the N-person controls described above for essential components of the launch control process also make less likely the unauthorised modification, replacement, theft, or destruction of these components. Because a loss, or loss of effectiveness, of any such component may inhibit weapon release, these N-person control measures also support the denial of service policy.

5.3 Information Disclosure Policy

As for the Aegis information disclosure policy, this policy is based on well-established DoD regulations and is directed at protecting classified information from individuals lacking sufficient clearances. Among the kinds of information of concern for the NC3 system are plans and contingencies for weapon deployment, and current status. The latter may include current capabilities, information about deployments in progress, and heightened states of operational preparedness. This information is protected by a variety of physical, procedural, and communications security measures.

5.4 Information Modification Policy

This policy is a subordinate policy whose objective is primarily to support the NC3 weapon release policy and denial of service policy. For example, if release orders are subject to unauthorised modification prior to being carried out, then it may be possible to subvert the intent of the release authorities, causing an unauthorised release. Similarly, to the extent that information is used as an enabling element in the launch control process, an unauthorised information modification could inhibit release, resulting in an unauthorised denial of service. Weapons orders, plans, and other types of release-governing information are protected against unauthorised modification by a variety of communications, physical, and procedural security measures including the N-person control procedures described above.

5.5 Policy Summary

The security policy objectives for the NC3 scenario include unauthorised disclosure and modification of information, unauthorised release of weapons, and unauthorised denial of service. Authorisation to access or use these resources is contingent on clearances, roles, separation of duty, split knowledge, N-person control, and source authenticated inputs.

6 Government Procurement Document Preparation

This scenario is concerned with the security policies associated with the government procurement process, primarily as they affect the government participants. The Computer-aided Acquisition and Logistics Support (CALS) program [21], is an ambitious attempt to automate much of this process in the future, as well as other activities supporting the design, manufacture, and logistical support of systems used by DoD. Unlike the previous two scenarios, which are based on existing operational and system requirements, this scenario is based on hypothetical future requirements extrapolated from fragmentary published descriptions of the

CALS program [21, 9, 10]. It concentrates on the preparation, approval, and release of Requests for Proposals (RFPs), allocation of government funds and manpower, and evaluation of competing bids.

6.1 Information Disclosure Policy

Much of the information associated with the procurement process may be sensitive with respect to disclosure. The procured items may have specifications that are classified. To ensure fair competition among bidders, information about the contents of RFPs under development must not be "leaked" prematurely to prospective bidders. There may also be information which is considered proprietary to a particular vendor. Dissemination controls (e.g., NOFORN, NOCONTRACTOR) and export control restrictions may need to be enforced.

6.2 Information Modification and Release Policy

The RFP development process consists of a sequence of draft, approval, and release phases. Among other purposes, these phases serve to prevent unauthorized procurement documents (e.g., erroneous RFPs and contracts) from resulting in unauthorized expenditure commitments of government funds and manpower resources. At each stage in the RFP development process, approval must be obtained prior to proceeding to the next stage. Furthermore, authority to submit, modify, or approve procurement documents at various stages is reserved to individuals who have been assigned particular roles. To the extent that procurement documents are kept on-line, controls are required to ensure that only appropriate individuals are able to update or modify information at each stage.

RFP development is initiated by a technical team whose members are authorized to generate a statement of work (SOW). Before an RFP can be generated, the SOW must be approved by management, and procurement funds must be allocated. The SOW is then forwarded to the contracts department, whose personnel are authorized to generate an RFP. The RFP must be approved by the legal department and approved for release by a contracting officer. As part of the release process, an authenticating code may be attached to assist bidders in verifying the authenticity of the RFP prior to committing their own resources for bid development.

6.3 Other Constraints

The roles held by individuals may be subject to role exclusion constraints. For example, members of the technical review team for bid evaluation may be prohibited from participating on the cost review team; this can be viewed as a form of separation of duty. Furthermore, they may also be forbidden from finding out about contents of the cost portions of bids. It may also be the case that an individual who has had access to a contractor's bid containing proprietary information may be forbidden from accessing a competing contractor's proprietary information for a set period of time.

6.4 Policy Summary

The security policy objectives for this scenario include preventing unauthorized disclosure of information, unauthorized modification and release of information, and unauthorized expenditure commitments of funds and manpower. Resource usage authorization is based on clearances and dissemination controls, roles, role exclusion constraints (including separation of duties), operation sequencing constraints and source authentication.

7 Observations

An analysis of the scenarios studied, including those outlined above, leads to a number of observations.

- Access control according to clearances or roles appears to be a fundamental aspect of each scenario. In particular, access control to protect information from both unauthorised disclosure and unauthorised modification was an element of every scenario. In addition, numerous other role-based access controls regulating use of resources other than information (e.g., weapons, financial assets) were found.
- Infrastructure support, particularly in the form of communications, identification and authentication, and auditing services, is likely to be applicable, independent of policy objectives. The extent of applicability depends on the geographic distribution of the organisation and the extent of its reliance on automation.
- Separation of duty constraints were found in several scenarios. This suggests that separation of duty is a well-established general principle that is widely employed when an organisation is reluctant to entrust unilateral control over a resource to any single individual. Furthermore, separation of duty requirements were sometimes accompanied by operation sequencing requirements. In some military environments, however, the principle of separation of duty may conflict with the need to ensure that, at all times, at least one individual (e.g., a commanding officer) has *sufficient authority* over resources to carry out an assigned mission successfully.
- Each scenario encompasses a unique combination of policy elements. No clear-cut patterns emerged to distinguish the policies for tactical scenarios, as a group, from those for non-tactical and commercial scenarios. However, because responsibilities must be rapidly and flexibly reassigned following combat casualties, tactical policies may tend to rely more heavily on fluid personnel authorisation methods including delegation of authority.
- Source authentication or non-repudiation requirements stipulating that personal or electronic signatures accompany data or permission to act, (e.g., military orders) appear to be widespread.
- "N-person rules" requiring teams of people to act simultaneously (or nearly so), and split knowledge requirements, were not common. This may be because their implementation is too costly or cumbersome to be used on a routine basis unless the resources being protected are extremely critical, as in the case of nuclear weapons.
- Numerous *requirements* related to denial of service, including requirements for reliability, survivability, and performance were encountered. However, few denial of service *policies* (as defined above) were identified; such policies govern the authority of particular individuals to use or operate on resources in ways that may deny use of those resources to otherwise authorised individuals. Several explanations for this result can be posited. First, denial of service remains an ill-understood problem, and denial of service policies remain difficult to identify definitively. Second, the security policy definitions used in this study deliberately exclude from consideration a variety of critical requirements that are commonly treated as security policy manifestations [27, 7, 22]. Third, primary threats to assured service in tactical systems include electronic warfare and the destruction of combat assets by the enemy. Threats of this nature are more naturally addressed by military tactics and improved equipment capabilities than by computer security technology, and were consequently deemphasised in the study.

These observations suggest that there exists a core set of security policies and policy elements that merit support in computing systems intended for a broad range of tactical, non-tactical, and commercial applications. This policy core includes protection of information from unauthorised disclosure and modification, role-based access control, role exclusion rules, (e.g., static separation of duty), delegation of authority, and operation sequencing.[4] The core also includes identification and authentication, auditing, and reliance on secure communications, especially to support source authentication and non-repudiation requirements.

These observations also support the contention that the TCSEC requirements are incomplete in comparison with secure tactical computing needs. A number of other security policies beyond confidentiality are integral to the contexts in which tactical systems are used, and it appears that tactical systems should be capable of providing some degree of automated support for these policies. Determining the extent to which policy support can be automated usefully, especially when possible system failure modes are taken into account, will require a significant level of continuing research addressing both human factors and systems engineering issues.

The authors feel it unlikely that automated mechanisms designed primarily for TCSEC requirements are well-suited to support these other policies. (Similar sentiments have been published elsewhere [11, 16]. See [4, 5] for a different perspective.) On the other hand, it appears that the information disclosure policies toward which TCSEC requirements are targeted remain crucial to tactical systems. Consequently, computing systems designed for a broad range of tactical applications should be minimally capable of satisfying TCSEC requirements in addition to supporting other organisational security policies.

8 Summary and Future Work

This paper has summarized the results of a study intended to identify security policies and common policy elements that may merit support in systems designed for tactical applications. While each analysed scenario appears to encompass a different combination of policy elements, the study suggests that these combinations may share a common policy core. This offers the hope that by supporting this common core, a single, configurable system may be able to support a wide variety of application specific security policies in the tactical, non-tactical, and commercial realms.

While a number of previous research papers have discussed table-driven, rule-driven, or otherwise configurable systems that may support multiple policies [14, 1, 17, 26, 4, 5], the feasibility and assurance potential of such systems remains an open research question; much more work is needed before a definitive answer can be put forth. Toward this end, functional requirements for a prototype system to support the policy core have been developed, and high-level design activities have been initiated. Follow-on plans include implementation of the prototype and an assessment of the applicability, effectiveness, and assurance of its enforcement mechanisms.

References

[1] Abrams, M.D., Eggers, K.W., La Padula, L.J., Olson, I.M., "A Generalized Framework For Access Control: An Informal Description," Proceedings of the 13th National Computer Security Conference, Washington, D.C., October, 1990.

[2] Baldwin, R.W., "Naming and Grouping Privileges to Simplify Security Management in Large Databases," Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1990.

[3] Bell, D.E., La Padula, L.J., "Secure Computer Systems: Unified Exposition and Multics Interpretation," Technical Report No. ESD-TR-75-306, Electronics Systems Division, AFSC, Hanscom AF Base, Bedford MA, 1976.

[4] Bell, D.E., "Lattices, Policies, and Implementations," Proceedings of the 13th National Computer Security Conference, Washington, D.C., 1990.

[4]Although dynamic separation of duty [19, 23, 12], is not discussed above, we include it as an extension of operation sequencing.

[5] Bell, D.E., "Putting Policy Isomorphisms to Work," Report 355-D, Trusted Information Systems, Glenwood, MD, November 1990.

[6] Clark, D.D., Wilson, D.R., "A Comparison of Commercial and Military Computer Security Policies," Proceedings of the 1987 IEEE Symposium on Security and Privacy, Oakland, CA, 1987.

[7] Gasser, M., *Building a Secure Computer System,* Van Nostrand Reinhold Co., New York, NY, 1988.

[8] Gersh, J., Private Communications, Johns Hopkins University Applied Physics Laboratory, October 1989, January 1990, August 1990, February 1991, June 1991.

[9] Gorham, Jr., W.C., "Data Protection Requirements in Computer-Aided Acquisition and Logistics Support," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.

[10] Gove, R.A., Friedman, A.R., "A Structured Risk Analysis Approach to Resolve the Data Protection and Integrity Issues for Computer-Aided Acquisition Logistics Support (CALS)," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.

[11] Graubart, R., "On the Need For a Third Form of Access Control," Proceedings of the 12th National Computer Security Conference, Baltimore, MD, October, 1989.

[12] Karger, P., "Implementing Commercial Data Integrity with Secure Capabilities," Proceedings of the 1988 IEEE Symposium on Security and Privacy, Oakland, CA, 1988.

[13] Karp, B.C., "The CALS Data Protection and Integrity Industry Working Group," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.

[14] La Padula, L.J., "Formal Modeling in a Generalized Framework for Access Control," Proceedings of the Computer Security Foundation Workshop III, June 1990.

[15] Mayer, F.L., "Security Controls for an Automated Command and Control Information System (ACCIS): Baseline Definition," Report 201, Trusted Information Systems, Glenwood, MD, May 1989.

[16] McCollum, C.J., Messing, J.R., Notargiacomo, L., "Beyond the Pale of MAC and DAC - Defining New Forms of Access Control," Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1990.

[17] Miller, D.V., Baldwin, R.W., "Access Control by Boolean Expression Evaluation," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.

[18] Murray, W.H., "Data Integrity in a Business Data Processing System," Report of the Workshop on Integrity Policy in Computer Information Systems (WIPCIS), Waltham, MA, October 1987.

[19] Nash, M.J., Poland, K.R., "Some Conundrums Concerning Separation of Duty," Proceedings of the 1990 IEEE Symposium on Security and Privacy, Oakland, CA, May 1990.

[20] National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria,* DOD 5200.28-STD, December 1985.

[21] O'Neal, S.A., "CALS: Enabling Process Improvement," Signal, September 1989, pp. 41-44.

[22] Pfleeger, C., *Security in Computing,* Prentice-Hall International, Inc., Englewood Cliffs, NJ, 1989.

[23] Sandhu, R., "Transaction Control Expressions For Separation of Duties," Proceedings of the Fourth Annual Computer Security Applications Conference, 1988.

[24] Sterne, D., Branstad, M., Hubbard, B., Mayer, B., Badger, L., Wolcott, D., "Security Policies for Tactical Environments: A Research Study," Report No. 353, Trusted Information Systems, Glenwood, MD, November 1990.

[25] Sterne, D., "On The Buzzword 'Security Policy'," Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1991.

[26] Thomsen, D.J., "Role-based Application Design and Enforcement," Proceedings of the Fourth IFIP Workshop on Database Security, Halifax, England, September 1990.

[27] Unisys, Camarillo, CA., and Trusted Information Systems, Inc., Glenwood, MD., SDI Battle Management Security Issues: A Preliminary View, TM-8361/000/00, February 1987.

ANOTHER FACTOR IN DETERMINING SECURITY REQUIREMENTS FOR TRUSTED COMPUTER APPLICATIONS

David Ferraiolo
NIST
Building 224/A241
Gaithersburg, MD 20899

Karen Ferraiolo
Grumman Data Systems
2411 Dulles Corner Park
Herndon, VA 22071

Computer systems take on security requirements that are unique to the operational characteristics and needs of their application. These requirements can be applied on an individual basis in reducing operational risk. Methods exist to determine security requirements per DoD 5200.28-STD [3] by calculation of a risk index [4], [6]. This risk index is used to determine an appropriate level of trust (criteria class) per DoD 5200.28-STD, which is then used to define a set of security requirements. However, the resulting security requirements imposed on some systems by DoD 5200.28-STD can be overly restrictive, in need of interpretation, or in many cases, non-applicable. The purpose of this paper is to provide insights into determining appropriate security requirements for applications within a specified criteria class. Observations depend to a great extent on the system's user interface, considered as an additional environmental condition.

Introduction

The intent of a computer application is to provide an organization with information processing capabilities in support of its specific mission or goals. It has become apparent that many of the security concepts defined by DoD 5200.28-STD do not directly apply in a general manner to all trusted computer applications. Some of the security features and assurances of DoD 5200.28-STD may be overly restrictive, others in need of interpretation and in some cases, are not applicable at all. This is because the six criteria classes that make up DoD 5200.28-STD, at least in part, assume a user environment with all the risks associated with that of a full-capability general purpose operating system. For many applications however, user capabilities are more restrictive than that of an operating system. Associated with these capabilities is a lower relative risk that coincides with the constrained ability for the user to influence the underlying processing environment.

Associated with each system is a User Interface Set (UIS). A UIS is a collection of processing capabilities provided to the users of the system. These capabilities include system prompts, menus, transactions, utilities, privileges, and operations. The UIS can provide for or preclude users from the following capabilities: execution of programs and transactions; creating and editing messages, documents, and files; creating, compiling and linking application or system programs. At a higher abstraction, a UIS can support an organization's security policy such as restricting individual users or groups of users to specific capabilities, functions and resources. For example, within a hospital system, a doctor may be provided with the capability to perform diagnoses, order tests, and prescribe medicine, while at the same time be prevented from directly performing updates or queries within the financial database.

For a large class of applications, the UIS may define a finite set of possible data accesses. The system's UIS constrains users by enforcing a template of capabilities. This template restricts users to the extent that the system can be viewed as a set of predefined resources (applications, communication links and user groups having specific capabilities). The security attributes which need to be associated with these resources can be defined by design specification. Because of these fixed resource attributes and the absence of a programming environment, security design techniques that are normally not acceptable for general purpose systems can be applied to meet specific security feature and assurance needs. For example, a peculiarity (with respect to DoD 5200.28-STD) that results from the stable functionality of many embedded computer systems, is the ability to allow access control decisions to be unambiguously established during system design time rather then having to be computed at run time. This is known as the binding of processes and data accesses, or simply early binding, a concept that is described in [7] and further exemplified by an access control triple described in [2]. In the context of DoD 5200.28-STD mandatory and discretionary access controls, subjects are thought of as representing people or the programs that act on their behalf, and objects as representing data or their files. However, in many embedded applications some subset of all the objects are not accessible to human users but are accessible only to the system hardware and software processes - these processes do not act as surrogates for users.

A good example is represented by a Regency Net (RN) terminal. The RN terminal is part of a tactical command and control system developed during the mid and late eighties. Although all users would be cleared for all information and belong to a single user group, security requirements were defined in respect to 1) the flow of data among multi-level resources, 2) the preservation of the integrity of critical data, and 3) the denial and delay of the delivery of critical messages. The concept of a reference monitor and security kernel was interpreted in order to ensure a high level of trust. The RN security kernel consists of an Initializer, to establish the CPU's initial secure state, and the Virtual Machine Monitor (VMM). Because the RN functionality is severely restricted, all subject-object access configurations are bound in the CPU Kernel code such that only those configurations which are both secure and functionally required are possible. The VMM is an extraordinary reference monitor in that it does not compute secure states, as a conventional reference monitor. It implements secure data flows directly rather than acting as an intermediate computational abstraction.

Another application dependent concept is captured by the Clark-Wilson [2] integrity model. The Clark-Wilson model defines an access control triple as the binding of a userID, transaction procedure (TP), and a set of constrained data items (CDIs). This binding indicates not only the ability to specify which users can access which executable program images (as is natural to normal DoD 5200.28-STD discretionary access control), but also implies that these executable program images (TPs) possess privileges in isolation from their invoking user.

Determining Environmental Risk

Defining meaningful computer security requirements for applications has not been a straightforward process. To help improve this situation, the National Computer Security Center (NCSC) published two documents, Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments and its associated Technical Rationale [4]. These two documents provide guidance for choosing an appropriate criteria class per DoD 5200.28-STD, by calculating a risk index based on the system's operating environment. The risk index is partially formulated by comparing the clearance of the least cleared user of the system to the highest classification of information to be processed by the system. The greater the difference between the clearance of the users and the classification of the information on the system, the greater the risk index and the greater the degree of trust that is required. Another environmental condition considers whether the personnel developing the application are authorized access to Secret information (or to the highest level of information to be processed by the system if the information classification is less than Secret). If not, the requirement is for a higher criteria class in order to compensate for the additional environmental risk developers impose on the delivered system.

It has been observed that not all potential risk associated with a system is due to the difference of the clearances of system users and the classification of information and the development environment. Risk may also result from other environmental factors. Another method, using other environmental factors to calculate potential risk has been developed by the Naval Research Laboratory (NRL) [6]. This report provides a more sophisticated approach for calculating risk, taking into account the environmental conditions of CSC-STD-003-85 as well as user processing capabilities and communication paths.

The Need for More Guidance

Both [4] and [6] define security requirements to the granularity of a predefined DoD 5200.28-STD criteria class. In the world of trusted computer applications, seldom have the calculated features and assurances of a criteria class of DoD 5200.28-STD defined a complete and essential set of applicable security requirements. Although this granularity may be at a reasonable level for products that are developed to be general purpose in nature (with no specific application in mind), for many applications minimal user capabilities can be ensured. Applicable security requirements can be defined (at least in part) in terms of the way a human user is intended to interact with the processing environment.

The premise of this paper is that, even though two systems may be defined as having the same risk index, and subsequently would require the same criteria class, applicable security features and assurances associated with these systems can vary significantly.

The Range of the Flexibility of User Interface Sets (R-FUIS)

It is suggested here that there is another significant environmental element that should be considered in determining information security requirements: the Flexibility of the User Interface Set (FUIS). As the flexibility provided through these interfaces increases, so does the risk that a user can influence and undermine the security preserving flow of information. This is regardless of whether the objective of an organization is to maintain the confidentiality of classified information, protect the privacy of individuals, ensure human safety, prevent fraud, or prevent unauthorized modification of educational records.

In order to consider the FUIS in the calculation of security requirements for applications, the FUIS must be measured in some way. The concept of a range in the flexibility of user interface sets (R-FUIS) is introduced. In theory, all systems fall somewhere on the R-FUIS. The relationship between these systems is such, that as systems progress on the range from left to right, applicable security features and assurances (requirements) appear that were not present prior to that point, until a point is reached where all features and assurances of DoD 5200.28-STD are present for a defined level of trust. Systems that fall to the extreme left have the most restrictive interface sets, and have the smallest subset of DoD 5200.28-STD requirements, while systems that fall to the extreme right are considered to have the most flexible interface sets and the most DoD 5200.28-STD requirements. Unlike the Risk Index, The R-FUIS represents a continuum where there is potentially an infinite number of possible points at which a system can be plotted. What is significant about the plotting of a system is where it falls relative to where other systems would fall. All systems can be plotted at some point on the R-FUIS. Depending on where systems fall, observations can be made as to security characteristics and requirements associated with that point. Moving from left to right along the continuum, the R-FUIS accounts for an extreme with no user interface; further along it accounts for a single user system, still further, multiple users but of a homogeneous nature (same role). Beyond the mid point, there are considerations for multiple users each belonging to a specific user role, while at the extreme right individual users with individual needs and privileges to access information are taken into account. (Instances of a role can include: a Doctor or Nurse within a hospital system; a Loan Officer or a Teller within a banking system; or a Traffic Analyst or Cryptanalyst within an intelligence system.) The concept of the R-FUIS is illustrated in Figure 1 below.

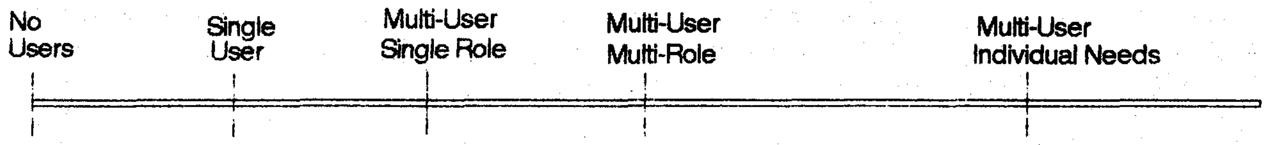


Figure 1. The Range in the Flexibility of User Interface Sets

A R-FUIS can be associated with each criteria class of DoD 5200.28-STD. The result is a two dimensional view of DoD 5200.28-STD, where there are 6 rows each representing a criteria class with the associated R-FUIS representing the range of applicable security requirements for that criteria class. An appropriate criteria class can first be determined through the use of current environmental guidelines [4], [6]. The position of the system on the R-FUIS for the criteria class can then provide insight as to applicable security requirements for the system.

The R-FUIS ranges from the most primitive or restrictive interface set, such as that of a black box with no user interface, to the most flexible interface, such as the full capabilities of a general purpose operating system. Both of these systems may process the same type and classification of information but because of the extreme differences in the FUIS, security requirements will differ greatly. The black box can be thought to have inherent security protection such as the inability of a human to alter its processing (except by physically removing its chassis and reprogramming it). However, a programming environment does not come as part of the system. It would need to be reprogrammed on another environment and down-loaded to the black box. On the other hand, the operating system supports the ability to create executable programs and alter existing ones. With the operating system interface, the following risks exist: the potential for introduction of a trojan horse, trap door, or virus; a program that mimics the operating system software and steals passwords; or the alteration of security relevant software. All these risks are a result of an operating system's natural user interfaces, while none of these risks are associated with the black box.

In order to reduce operational risk, security requirements are imposed throughout the system development cycle. These requirements must then be evaluated to ensure a secure operating environment. When a certification is performed for an application to operate in a specific environment, the certification should be an evaluation of the applicable security requirements associated with that system type. Because this evaluation would be conducted against some subset of requirements of a specified criteria class, it may not be appropriate to assign a criteria rating to the system, but instead indicate that the system mitigates known security risk, and is known to implement some list of security features and some level of assurances.

Defining Applicable Security Requirements

The R-FUIS can be subdivided in several ways depending on the UIS associated with the various types or categories of systems. By subdividing the R-FUIS into various types of systems and defining the security characteristics belonging to each of the types, the R-FUIS can be used to provide insight in the definition of security requirements. It is acknowledged that the use of the R-FUIS does not provide an absolute solution to defining security requirements for trusted applications. However, a widely agreed upon definition of the R-FUIS could provide guidance and establish precedence as to applicable security requirements that could be used from project to project, making the definition of applicable security requirements less of a subjective process.

By continuously subdividing the R-FUIS into smaller and more numerous pieces, the R-FUIS will be more helpful in the definition of security requirements. However, it is not the intent here to define an extensive list of possible types of computer systems. Instead, four types of systems are described and plotted on the R-FUIS to demonstrate how the R-FUIS can be used. By plotting a system on an even sparsely defined R-FUIS, guidance can be provided as to the system's applicable security requirements.

Observations on the R-FUIS

In the examples presented in this section, security features of DoD 5200.28-STD are described as they apply for each type of application. Figure 2 below summarizes these observations, providing one view of the R-FUIS.

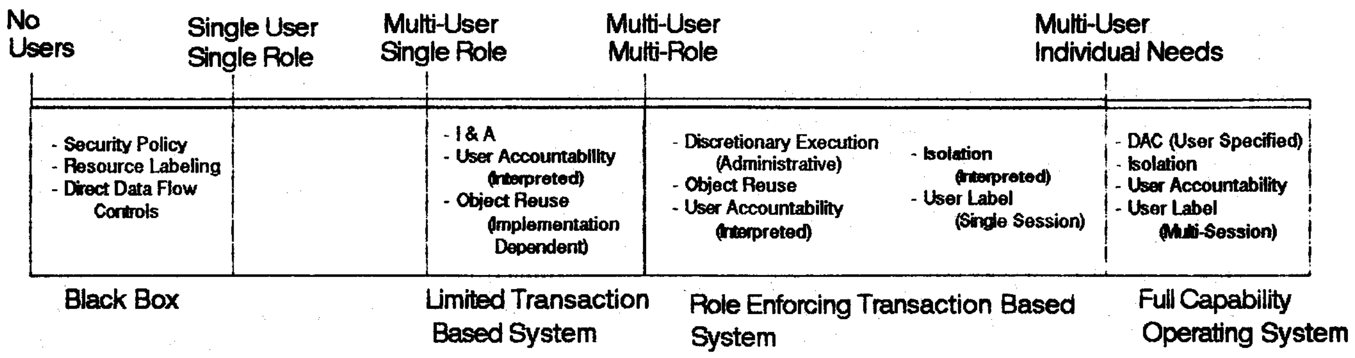

No Users
Single User Single Role
Multi-User Single Role
Multi-User Multi-Role
Multi-User Individual Needs
- Security Policy
- Resource Labeling
- Direct Data Flow Controls
- I & A
- User Accountability (Interpreted)
- Object Reuse (Implementation Dependent)
- Discretionary Execution (Administrative)
- Object Reuse
- User Accountability (Interpreted)
- Isolation (Interpreted)
- User Label (Single Session)
- DAC (User Specified)
- Isolation
- User Accountability
- User Label (Multi-Session)
Black Box
Limited Transaction Based System
Role Enforcing Transaction Based System
Full Capability Operating System

Figure 2. Example Systems Mapped onto the R-FUIS

<u>Black Box Systems</u>

For the most restrictive systems, which will be typed Black Box, many of the security features and assurances of DoD 5200.28-STD are not applicable. For the Black Box system, no humans have the ability to directly influence (read, or write) its objects. These systems are usually components incorporated to perform one or more specific control functions within a larger system. A Black Box system can be thought of as a "closed" system that contains only embedded processes where none of these processors contain a direct man-machine interface. In fact, in many applications a Black Box provides specialized services to a larger system which is totally transparent to human users. Although a Black Box system does not support the direct needs of human users it still may be trusted to perform a vital processing function. Military Black Box applications are numerous but they can include civil and commercial applications as well. For example, the routing of mail, aircraft avionics, robot control, and transportation switching devices. What is significant is that the execution of the controls of the device can be assumed to be free of human interaction.

Obviously for Black Box systems direct user related features such as identification and authentication, and user accountability are not applicable. In addition, making access control decisions based on the identity of or an attribute associated with a direct user does not make sense within a Black Box environment -- no Discretionary or Mandatory Access Controls with respect to the UIS. Also, there is not a requirement for a trusted path between a system user and the TCB.

The applicable security features can be viewed as the smallest subset of DoD 5200.28-STD requirements. These security features are relevant for all systems of this specified R-FUIS and the specified criteria class. All systems that fall to the right of the black box will include these fundamental features in their list of security requirements. Probably the most fundamental of all security requirements is that of a security policy. It is the security policy that defines what it means for a system to be secure. All other security features and assurances are present only in support of that policy. This policy may ensure that information of varying levels does not get mixed while in the local system. Because there exists a security policy there must be an associated mechanism to implement the policy. For many Black Box applications, controls are flow-oriented where the policy is preserved through flow decisions that could be considered at design time rather than at run time. Object reuse more than likely would not be applicable. Pools of previously used memory are not available for subsequent scavenging.

Because there is no concept of application software as opposed to system software, there is a de-emphasis on the need for isolation techniques, such as domains of execution. Strong physical and procedural controls can be applied during system development to ensure an execution environment that is free of malicious code. Tools can be applied to ensure all flows are security preserving. Lastly, the absence of a user interface goes a long way in ensuring that the secure environment stays secure.

Limited Transaction Based Systems

The next type of system that will be described is a Limited Transaction based system. A good example of a Limited Transaction based system is an Automatic Teller Machine (ATM). For these systems there is the presence of a man-machine interface, although it is quite limited. All users generally belong to one user group. Although this group may potentially be quite large, the users are constrained to a narrow set of processing capabilities and all perform the same functions.

For example, there may be a menu where selections can be made via a simple interface device such as a numeric key pad. For a Limited Transaction based systems, users are precluded from accessing information other than through well defined inter-related sets of processes known as transactions. For systems of this type, subject-object access configurations can be pre-specified and bound in such a way that only those access configurations which are both secure and functionally required are possible. Authorized users are first identified and then given "select" access to a limited set of transactions, which in turn have access privileges to information. By making a selection on a menu, a transaction is started and a specified and controlled set of activities occur. This transaction will access and manipulate specific files based on the type of transaction being invoked. The only access to information is defined by specification and determined during the system design.

For many Limited Transaction Based Systems most of the objects are not accessible to human users but are accessible only to the system hardware or software processes. It is the data and the flow of information associated with these processes that are security relevant rather than humans accessing information. There may be some number of secure data communications links where the information may be multilevel in nature. The system must be trusted not to mix information of a higher level with that of a lower level where it would then be perceived as being of the lower level (this is the mandatory policy). While supporting secure links, this type system could also support a link that is not secure (unencrypted) which would have to be considered unclassified. Although there exists multiple users each belongs to the same role and would possess the same security clearance. The users security attributes would be considered fixed for which data would flow accordingly.

Identification and authentication mechanisms are generally used for accountability purposes alone. Discretionary access controls (per DoD 5200.28-STD) do not apply in the sense that user's can specify what other users have access to the files. For the ATM example, the user's Personal Identification Number (PIN) may be used as a parameter within a transaction for the purposes of retrieving the correct account record. No capability exists for users to grant or revoke privileges other than through disallowing access to the system.

Role Enforcing Transaction Based Systems

A Role Enforcing Transaction based system is similar in many ways to a Limited Transaction based system in that the access to information is granted or configured in terms of a process or transaction ID. For this type system all users have a proper clearance and need-to-access within their role. Therefore a user security level can be assumed (no need to specify security session level) and as with the Limited Transaction based system the flow of data can be considered accordingly.

The access control mechanisms principally enforce the rigid concept of least privilege and not the richer mechanisms implied by discretionary access control. Role Enforcing systems restrict access to information based on the role a user chooses. A given user may have the ability to move from role to role if he is authorized, but the user can only take on one role at a time. A user would choose a role via a menu selection, at which point a validation would be conducted to ensure the user can take on that role. The user's identity is critical for both validating that his role is legal and accounting for his actions. The method of enforcing need-to-access is not implemented through a strict discretionary access control mechanism as defined in DoD 5200.28-STD, but rather through a series of mechanisms and characteristics of the system. First, a check is made as to whether a user can take on a selected role. If the user is granted access to the role, he is given execute access to a series of transactions that are presented to him. The user is presented only with a list of transactions that has been specified to support his role. After the selection of a transaction, the user specifies parameters associated with the transaction and hits a return key or clicks a mouse to effectively start the transaction. The transaction is deterministic, performing activities in support of the user's role. Individual users can be added and deleted to each role. Role membership is most likely centrally administered rather than at the discretion of the individual users. Role membership may be altered through administrative and procedural controls. The capabilities represented by each role would be static in nature and could not easily be changed.

A role enforcing system in many cases supports a type of mandatory (non-discretionary) access control policy. Consider the hospital example described above. The system may provide a physician with the capability to perform a diagnosis, prescribe medication, and add to a record of treatments performed on a patient. Here roles would be created to preclude the physician from giving away the capability to perform a diagnosis or prescribe medicine to a non-physician. It is also a mandatory policy that users are prevented from modifying the record of treatments maintained for each patient.

The deterministic characteristics of the system is an important consideration in maintaining an audit trail. UserID, time, transactionID, and transaction parameter entries, in many cases are sufficient in holding users accountable for their actions. However, any one transaction may invoke numerous processes across several platforms and access countless data items. To audit each successful access would provide an overwhelming amount of information.

Full Capability Operating System

The most complex of all human interface sets is associated with an operating system. A Full Capability Operating system supports many users simultaneously while at the same time enforces both a mandatory and discretionary access control policy with respect to users and information. Discretionary access control mechanisms allow users to specify, using their discretion, the access privileges other users have to the objects they own. Although discretionary access controls are intended to be the principal means of enforcing need-to-access, these controls are inherently insecure. Because of a real possibility of users introducing malicious software, a more reliable mechanism than a discretionary access control mechanism must be provided, namely mandatory label-based access controls. These mandatory access controls are typically provided through the enforcement of the rules of the Bell & LaPadula security model [1].

Within an operating system environment, mandatory security rules must consider fixed resources, the assumption of malicious software, users with different security clearances, and data of multiple security levels. Here a run time access control intermediary must be provided that enforces the rules of the Bell & LaPadula security model. This access control intermediary is based on the concept of a reference monitor and may be implemented as a security kernel, depending on the risk index calculated for the application. Before a subject is permitted to have access to an object, a run time check is performed to ensure that the proposed access conforms to the set of underlying security rules governing the system. The theory of security in an operating system is induced from an initial secure state and a demonstration of the preservation of security for every operation subsequently allowed by the reference monitor. In essence, the purpose of a reference monitor is to compute security states for the system.

To preclude the ability of a subject from having simultaneous read access to an object of a higher classification and write access to an object of a lower classification (where there is the potential for an illicit flow of information) a rule similar to the *-property must be enforced. The *-property requires the subject's security level to be equal to or lower than that of an object for which the subject is attempting to gain write access. Because the classification of the object can be lower than the highest security clearance of the user, a method must be provided to allow the user to establish a session level lower than that of his or her highest security clearance. If that user needs to read an object of a higher classification then the object to which the user just wrote, the user's session level must be raised to a level at least as high as the level of the object to be read.

Application software can be added or modified at any time during the operational life cycle of the system, and often is created by the very subjects to which the rules of the security policy apply. In order to provide a reasonable degree of assurance that applications software cannot by-pass or alter the security policy enforcing mechanisms, security mechanisms must reside in a separate and more privileged execution domain than that of the applications software.

Further, to preclude a malicious user from stealing an unsuspecting user's password through the creation of a program that mimics a legitimate password request, a trusted path must be provided. A trusted path would ensure a reliable communication channel between system users and security relevant software.

Conclusion

Computer applications range from a black box which has no direct system user, to a very flexible system supporting many human users simultaneously, where these users have the ability to create executable images of programs and share information on a discretionary basis. It is reasonable to believe that although these systems may have the same calculated risk index per [4], they should implement only security mechanisms that are applicable to their operational environment.

The R-FUIS (Range in Flexibility of the User Interface Set) has been introduced to provide insights to determining security requirements for a system based on characteristics of the application as well as other environmental conditions identified in [4] and [6]. It is acknowledged that the use of the R-FUIS will not provide an absolute solution to determining security requirements, but it is our hope that the determination of applicable security requirements may become more of a methodology.

By further defining the R-FUIS innovative security design techniques can be uniformly applied across new secure application development efforts. This definition can be provided through the consideration of existing and future secure application development cases studies. The result would be new and increasing numbers of innovative security techniques uniformly applied within appropriate (better defined) security environments. Through a peer review process new security techniques can be accepted as legitimate methods in combating environmental risk. The existence of criteria, criteria interpretation, and guidelines should never result in the stifling of new and innovative approaches for applying security within our systems.

References

[1] Bell, D.E., LaPadula, L.J., Secure Computer Systems: Mathematical Foundations, ESD-TR-73-278, Vol. I, Electronic Systems Division, Air Force Systems Command, November 1973.

[2] Clark, D.D., Wilson D.R., "A Comparison of Commercial and Military Computer Security Policies," Proc. on Security & Privacy, Oakland, CA., pp. 184-194, IEEE Computer Society, April 27-29, 1987.

[3] Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, Department of Defense 5200.28-STD, December 1985.

[4] Department of Defense Computer Security Center, Computer Security Requirements - Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, CSC-STD-003-85, National Computer Security Center, June 1985.

[5] Goguen, J.A., Meseguer, J., "Security Policies & Models," Proceedings 1982 Symposium on Security & Privacy, Oakland, CA., pp. 11-20, IEEE Computer Society, April 1982.

[6] Landwehr, C.E., Lubbes, H.O., "An Approach to Determining Computer Security Requirements for Navy Systems," NRL Report 8897, Naval Research Laboratory, May 1985.

[7] Norton, W., Reider, L., "Computer Security in Regency Net," DRAFT.

APPARENT DIFFERENCES BETWEEN THE U.S. TCSEC AND THE EUROPEAN ITSEC

Dr. Martha A. Branstad
Dr. Charles P. Pfleeger
Trusted Information Systems, Inc.
3060 Glenwood, MD 21738

Dr. David Brewer
Gamma Secure Systems, Ltd.
Diamond House
149 Frimley Road

Mr. Christian Jahl
Mr. Helmut Kurth
IAGB Software Technology
EinsteinstraBe20
D-8012 Ottobrunn

The U.S. *Trusted Computer System Evaluation Criteria*, called the TCSEC [TCS85] which was first published in 1983 and revised in 1985, has become an accepted standard for the evaluation of trusted systems. Not only is it used in the U.S. for evaluations by the National Computer Security Center (NCSC), it has also been adopted by NATO for the evaluation of systems for use in NATO installations. More recently, in May 1990, a group of four nations, France, Germany, the Netherlands, and the United Kingdom, produced a first draft of its *Information Technology Security Evaluation Criteria*, called the ITSEC [ITS90]. The ITSEC shows clearly that the thinking of the computer security community has been heavily influenced by the TCSEC, but the ITSEC also addresses some issues in ways that are very different from the TCSEC. A meeting was held under the sponsorship of the European Commission in Brussels on September 25-26, 1990, at which members of the four nations discussed their reactions to comments received since the publication of the ITSEC and presented their opinions of changes that should be made to the ITSEC.

As a method of understanding the ITSEC more completely, it was analyzed to determine the impact that compliance with an F5/E5 rating would have upon a B3 targeted system that is under development. This analysis led to a discussion with ITSEC authors from both Germany and the United Kingdom that helped to clarify many questions concerning specific wording and concepts of the ITSEC and its relationship with the TCSEC. It should be noted that the views presented here are the authors' and not official statements from the various organizations with which they are affiliated.

BACKGROUND

TCSEC Overview

Briefly, the TCSEC establishes six levels of evaluation: C1 and C2 provide discretionary access control only, B1, B2, B3, and A1 provide both discretionary and mandatory access control. Beginning at B2 and progressing to B3 and A1, the requirements for assurance — measures that inspire confidence that the implementation of the system truly and rigorously enforces its stated security policy — play a very significant part in the evaluation. Each TCSEC rating, called a digraph, is thus a combination of a particular set of features and a necessary minimum degree

Portions of the work reported in this paper were performed on DARPA contract #F30602-89-C-0125

of assurance. Since publication of the TCSEC, there has been discussion in the computer security community over the advisability of this bundling of features and assurance. There has also been considerable discussion regarding the predetermined collections of features represented by each digraph class; little room is available for the development and evaluation of a trusted system that had goals other than maintaining confidentiality.

ITSEC Overview

In consideration of these two concerns, the authors of the ITSEC chose to separate the functionality of a trusted system[1] from ratings of its assurance[2], and to expand the range of functionality. Each evaluated trusted system would be awarded two descriptors: one denoting the functionality the trusted system presents, and the second, denoting the assurance of correct implementation of that functionality. Currently, there are ten exemplary predefined functionality classes, F1–F5 and F6, F7, F8, F9, and F10. The classes F1–F5 correspond closely with functionality required at the TCSEC classes C1, C2, B1, B2, and B3[3], respectively. The five remaining predefined classes represent integrity, availability, data communications integrity, data communications confidentiality, and data communications integrity and confidentiality, respectively. A trusted system can be evaluated against more than one of these classes of functionality, if appropriate. There is also the potential for a developer to define a new class of functionality, if these classes fail to describe a particular trusted system adequately. Assurance is recognized as a combination of correctness and effectiveness. Six correctness ratings were defined as E1–E6; these combine with the judgement of effectiveness of the security functions and mechanisms. These assurance ratings were intended to correspond generally to the TCSEC assurance requirements for C1, C2, B1, B2, B3, and A1 trusted systems, respectively. Thus, given trusted systems might achieve ratings of F3/E2 or F4-F7/E4, for example.

Because the requirements for the F1–F5 and E1–E6 classes so closely resemble the TCSEC requirements, it is reasonable to try to identify points where ITSEC and TCSEC ratings coincide. The annex to Appendix A of the ITSEC lists the intended correspondences from the ITSEC to the TCSEC, that is, functionality/assurance combinations that are at least as strong as TCSEC digraphs. Table 1 shows these intended correspondences. The ITSEC criteria contain a number of requirements that do not appear in the TCSEC explicitly, and thus, according to the ITSEC, direct equivalence of evaluation levels is inappropriate.

---

[1] The ITSEC distinguishes between a "product" which is intended to be useful in a wide range of application environments, and "system" which is designed and built for the needs of a specific type of environment. The term "trusted system" is used in this paper to denote either a product or a system that is being evaluated under one of the criteria.

[2] The separate evaluation of functionality and assurance was first documented in the German II Security Evaluation Criteria [GISA89].

[3] TCSEC class A1 was omitted from this list because its functionality requirements are identical to those of class B3.

Table 1 Intended correspondence from ITSEC to TCSEC

ITSEC Class	TCSEC Class
F1/E2	C1
F2/E2	C2
F3/E3	B1
F4/E4	B2
F5/F5	B3
F5/E6	A1

However, it is also true that there are requirements in the TCSEC that were not replicated in the ITSEC. Thus, the correspondence of Table 1 does not work in either direction. Still, it is a good starting point for analyzing the differences between the two evaluation criteria.

TCSEC/ITSEC CORRESPONDENCE

As a method of understanding the ITSEC more completely, it was analyzed to determine what was involved in achieving compliance with an F5/E5 rating and what the impact would be upon a B3 targeted trusted system that is under development. First the ITSEC was examined to determine (a) how a trusted system could be evaluated as F5/E5 yet fail to meet B3, and (b) how a system could be evaluated as B3 yet fail to meet F5/E5. The intention of this analysis was first, to understand better the nuances of the requirement language, and second, to determine what additional work a developer would need to do in order to produce a system that met both criteria. After the identification of apparent differences, some of the TCSEC authors, some of the ITSEC authors, and some others met to determine if the apparent differences were really intended. Among the ITSEC authors, there were representatives both from Germany and the United Kingdom. The remainder of this report describes the outcome of that meeting. It should be noted that the participants at the meeting were presenting their own views of the sense of the groups of which they are a part.

Attributes of Trusted Systems that could Pass F5/E5 but Fail B3

The following sections present statements from the TCSEC and the ITSEC, followed by a brief statement of intention from the authors. Section or page number references are included. **Bold face** type is reproduced from the original; underlining is used to draw attention, but is an addition to the original text.

1. No DAC or DAC does not apply to all named objects.

TCSEC: §3.3.1.1 ...These access controls shall be capable of specifying, for each named object, a list of named individuals, ...

ITSEC: F5, p. 104 (§ Administration of rights) The system shall be able to distinguish and administer access rights between each user and/or user group and the objects <u>which are subject to the administration of rights.</u>

Discussion: The ITSEC drafters indicated that it was their intention to have this F5 requirement correspond to B3. Rework of the ITSEC wording could make the equivalency more evident.

The ITSEC drafters wanted to avoid the term "named object" since there has been some controversy about its meaning in the TCSEC. There is ambiguity in the phrasing of the ITSEC, however. The ITSEC drafters intended for this requirement to apply to all objects defined by and visible to users. In any system there are three classes of objects that might be subject to administration of rights: i) those that are defined by and visible to users, ii) those that are defined by the system and may be directly or indirectly visible to users, and iii) those that are defined by the system but used at a level below that at which access control policy is enforced. The objects of class iii) are not subject to the administration of rights but must be considered in covert channel analysis. Those of classes i) and ii) are subject to mandatory access controls. The objects of class i) are subject to discretionary access controls.

The ITSEC drafters acknowledge that they would like less restrictive language to apply to lower assurance levels. Progressively more stringent requirements for applicability of access control were desired as the assurance level rose within a given functionality class, but given the separation between functionality and assurance, it is very difficult for the ITSEC authors to impose such progressive requirements within one functionality class. The ITSEC authors are searching for a way to delineate those objects that must be subject to administration of rights; it may be that the categorization of class i, class ii), and class iii) above is a way to achieve this.

2. MAC does not apply to all resources directly or indirectly accessible by subjects external to the TCB.

TCSEC: §3.3.1.4 The TCB shall enforce a mandatory access control policy over <u>all resources</u>...

ITSEC: F5, p. 106 (§ Verification of rights) With each attempt by users or user groups to access objects <u>which are subject to</u> the administration of rights, the system shall...

Discussion: This is the same problem as above. The wording of the TCSEC is open to some interpretation (e.g., whether or not it is intended to apply to a system console). The intention of the ITSEC authors was to be equivalent to their perception of the meaning intended in the TCSEC.

3. Encrypted storage is not cleared before reuse.

TCSEC: §3.3.1.2 No information, <u>including encrypted representations of information</u>, ... is to be available to any subject that obtains access to an object that has been released back to the system.

ITSEC: F5, p. 107 (§ Object Reuse) All storage objects returned to the system shall be treated before reuse by other subjects, in such a way that no conclusions can be drawn regarding the preceding content.

Discussion: The distinction between the TCSEC and the ITSEC was intended. If encryption is judged adequate to protect data in transmission or storage, then it should also be adequate to prevent any determination of plaintext from ciphertext that may be obtained from a reused object.

4. Human readable labels are provided, but not at places specified.

TCSEC: §3.3.1.3.2.3 The TCB shall mark the beginning and end of all human-readable, hardcopy output.

ITSEC: F5, p. 106 (§ Administration of rights) The system shall mark human readable output with attribute values. The values of the attributes shall be determined according to the rules laid down in the system. Authorized users shall be able to specify the printable name of each attribute value and the location of the corresponding marking.

Discussion: The distinction between the TCSEC and the ITSEC was intended. It should be a matter of agreement between system user, system designer, and system security administrator precisely where the labels are placed.

5. The trusted path mechanism is not available for the user to change security level or to query the system about security level.

TCSEC: §3.3.2.1.1 The TCB shall support a trusted communication path between itself and users for use **when a positive TCB-to-user connection is required (e.g., login, change of subject security level).**

ITSEC: F5, p. 106 (§ Administration of rights) A user shall be notified immediately of any change in the security level associated with that user during an interactive session. The user shall be able at all times to display all the subject's attributes.

Discussion: The authors of the ITSEC have consciously tried to separate functionality requirements from mechanisms by which those requirements are implemented. They do not wish to be prescriptive of specific mechanisms in their requirements. However, without a trusted path in an F5/E5 trusted system, there is a possibility that an untrusted process could masquerade as the login process, thereby capturing a user's login and authentication date. This presents a security threat which, if included in the trusted system's Security Target (the actual baseline against which the system is evaluated, see Chapter 2 and page 63 of the ITSEC) would need to be identified and countered, for an E5 rating.

The ultimate difference here is that the TCSEC authors felt strongly enough about the need for a trusted path at the B3 level to mention it explicitly. In the ITSEC the issue is handled through the suitability of functionality and strength of mechanism requirements of assurance - effectiveness (§4.2.1 and §4.2.4). The trusted path is not an explicit requirement of the ITSEC, but it, or a similarly effective mechanism, would be needed to counter the threat of a masquerade of the login procedure. Explicit specification of implicit effectiveness requirements would lead to greater clarity of actual ITSEC requirements. This is another instance in which a low assurance class might not necessitate such a strong mechanism as the trusted path, which would be very appropriate at the higher assurance levels.

6. No identifiable reference monitor exists.

TCSEC:	§3.3.4.4 Documentation shall describe how the TCB implements the reference monitor concept and give an explanation why it is tamper resistant cannot be bypassed, and correctly implemented.
ITSEC:	no such explicit requirement exists
Discussion:	The identification of a TCB and implementation of protection through the reference monitor concept was seen by the ITSEC authors as being associated with specific security policies and prescriptive of particular mechanisms. On the other hand, the ITSEC authors recognize the desirability of the reference monitor concept and TCB in many instances. They intend to use the effectiveness component of assurance to exclude systems that fail to use the reference monitor concept when it would have been more appropriate than whatever approach the developers used. A need for greater specificity of the effectiveness requirements is recognized, but such specificity is difficult to achieve while maintaining the goal of policy generality. It is of course open for the person defining the security target for an ITSEC F5/E5 evaluation to mandate the use of particular types of mechanism, for example a reference validation mechanism implementing the concept of a reference monitor. Clearly, this then constrains the developer to follow a TCSEC-like approach.

7. TCB not appropriately structured

TCSEC:	§3.3.3.1.1 The TCB modules shall be designed such that the principle of least privilege is enforced... It shall make effective use of available hardware

to separate those elements that are protection critical from those that are not. **The TCB shall be designed and structured to use a complete, conceptually simple protection mechanism with precisely defined semantics. This mechanism shall play a central role in enforcing the internal structure of the TCB and the system. The TCB shall incorporate significant use of layering, abstraction, and data hiding. Significant system engineering shall be directed toward minimizing the complexity of the TCB and excluding from the TCB modules that are not protection critical.**

ITSEC: no such explicit requirement exists

Discussion: This issue is essentially the same as the trusted path issue explored above. All of these structuring requirements were seen by the ITSEC authors as prescribing mechanisms that would be very appropriate in many situations but might not be appropriate in all. Their intention is to treat this issue in the effectiveness section. It is likely that this issue will be addressed in a manual for evaluators, by way of example.

Observation: The TCSEC and ITSEC authors recognize these last two points as definite differences between the TCSEC and the ITSEC. If a developer wants to achieve both F5/E5 and B3 evaluations, the developer will want to plan to meet both sets of requirements. The ITSEC authors recognized that exact correspondence with the TCSEC was impossible within the ITSEC scheme. They have indicated that their intention was that trusted systems evaluated at the F5/E5 or the B3 level should yield equivalent assurance of enforcement of the defined security policy.

Attributes of systems that could pass B3 but fail F5/E5

1. Fail to provide a read-only access mode.

TCSEC: §3.3.1.1 These access controls shall be capable of **specifying, for each named object, a list of named individuals... with their respective modes of access to that object.**

ITSEC: F5 p. 104, (§ Administration of rights) It shall also be possible to restrict a user's access to an object to those operations which do not modify it.

Discussion: Since many commercial clients are concerned with controlling the ability of a user to modify information but are not concerned with whether the user can read the data, the existence of read-only access mode is deemed important.

2. Fail to provide labels for subjects and objects internal to the TCB.

TCSEC:	§3.3.1.4 The TCB shall enforce a mandatory access control policy over all resources... that are directly or indirectly accessible by subjects external to the TCB. These subjects and objects shall be assigned sensitivity labels...
ITSEC:	F5, p 105 (§ Administration of rights) In addition, the system shall provide all subjects and objects... with attributes.
Discussion:	The words as currently written do not convey the intended meaning of the ITSEC authors.

3. Fail to provide design documentation for non-TCB elements.

TCSEC:	no requirement
ITSEC:	§3.6.1.1.1 The sponsor shall provide the following documentation... structured description of the detailed design.
Discussion:	The ITSEC authors indicated that their intent was for design documentation to be required only for parts of the system critical to the enforcement of security.

4. Use a non-validated compiler.

TCSEC:	no requirement
ITSEC:	§3.6.1.2.2 b) The used compilers shall be **validated e.g., approved by an appropriate body**.
Discussion:	This requirement was discussed at the Brussels conference; it is expected that the requirement will be reworded.

5. Use a non-rigorous notation for the architectural design.

TCSEC:	no requirement
ITSEC:	§3.6.1.1.3 c) The architectural description shall use some form of rigorous approach and notation.
Discussion:	The concept is appropriate for reconsideration by ITSEC authors. At the Brussels conference, a number of inconsistencies were reported in Chapter 3 of the ITSEC. Rewording of requirements is likely.

6. Provide no mathematical analysis of design refinements.

TCSEC:	no requirement
ITSEC:	§3.6.1.1.4 c) **Mathematical reasoning shall be used to show that each hierarchical level is a refinement of the previous level.**
Discussion:	The intention of the ITSEC authors was to support traceability between levels of the design. The term "logical" is perhaps a better choice than "mathematical" to express the ITSEC authors' intent of supporting traceability.

7. Include functions with side-effects.

TCSEC:	no requirement
ITSEC:	§3.6.1.1.4 c) An analysis of the detailed design for side effects shall indicate that none exist and that no additional functionality is present which would allow the security mechanisms to be bypassed.
Discussion:	This distinction is both semantic and substantive. In some European evaluation circles, a "side effect" is something that a trusted function does which is security-relevant and which the function is not intended to do. In the U.S., "side effect" is used more broadly to mean any effect beyond the <u>defined functionality</u>. The narrower usage is consistent with the intention of the TCSEC as described under Security Testing (§3.3.3.2.1) as "their [testers'] objectives shall be: to uncover all design and implementation flaws..." The intention of the ITSEC authors was to prohibit side effects that could undermine the security policy enforcement. The difficulty in designing a completely side-effect free product is acknowledged. The ITSEC wording could be clarified.

8. Fail to map security functions to mechanisms.

TCSEC:	no requirement
ITSEC:	§3.6.1.1.4 b) It **[the specification document]** shall also map security functions to mechanisms and **functional units**. §3.6.1.1.4 c) It shall be shown that the security mechanisms provide the security functions stated in the security target.
Discussion:	This requirement was intentionally included by the ITSEC authors; however, it is anticipated that this requirement might be met by a level-by-level analysis as part of the philosophy of protection required by the TCSEC.

9. Fail to identify all non-local variables.

TCSEC: no explicit requirement

ITSEC: §3.6.1.1.4 b) **All variables used by more than one functional unit shall be defined at the lowest level of the specification and their purpose shall be explained.**

Discussion: This requirement was intentionally included by the ITSEC authors as an extension of the TCSEC.

10. Provide inadequate configuration management tools by (a) failing to illustrate item relationships or (b) failing to identify security relevant changes.

TCSEC: no explicit requirement

ITSEC: §3.6.1.2.2 b) All objects created during the development process, such as design documents, source code, and other dependent data shall be subject to configuration control. ... In the event of a change of any of these objects, the tools shall be able to identify all objects affected by this change. The tools shall support the determination of whether a change is security relevant.

Discussion: This requirement was intentionally included by the ITSEC authors as an extension of the TCSEC. Part (b) was not intended to be extreme; its intention was to force the developer to separate the code into a part that was security relevant and a part that was not. Changes to only the security relevant code were to be tracked; and any change to security relevant code was to be tracked.

11. Provide inadequate vulnerability analysis.

TCSEC: No general vulnerability analysis requirement exists, but sections 3.3.3.1.3 and 3.3.3.2.1 require covert channel analysis and penetration testing, respectively.

ITSEC: §3.6.1.1.4 b) The design vulnerability analysis shall determine any ways in which it is possible for a user of the TOE to deactivate, bypass, corrupt, or otherwise circumvent the security afforded by the TOE as configured by a security administrator.

§3.6.1.1.5 b) **The implementation vulnerability analysis shall determine any ways in which it is possible for a user of the TOE to deactivate, bypass, corrupt, or otherwise circumvent the security afforded by the TOE as configured by a security administrator, based on the source code. It shall identify covert channels.**

Discussion: The ITSEC authors recognize that defining what constitutes an adequate vulnerability analysis is difficult, especially for systems and products that span a collection of varying security policies. The authors intend to include more specific guidance in the manual for evaluators. For the present, in confidentiality-preserving systems, the authors' intent was that penetration testing and covert channel analysis suffice for a vulnerability analysis.

Comment: With respect to penetration testing, the ITSEC authors expect that the developer and the evaluators will be in a cooperative, not an adversarial, relationship. The developer will undoubtedly perform some amount of penetration testing; notes on the analysis required to hypothesize penetrations and the tests performed to validate the hypotheses will reduce the amount of work the evaluators need to perform for penetration testing.

Moreover, covert channel analysis is only applicable under certain circumstances, i.e., where the security policy concerns confidentiality and the threat of covert channel attack is included in the Security Target. Thus it may be better to move the covert channel analysis requirement (pages 57, 65, and 73 of the ITSEC) from Chapter 3 to the predefined functionality classes F4, F5, and F6.

12. Fail to use test coverage tools.

TCSEC: no requirement

ITSEC: §3.6.1.1.5 b) The test documentation shall contain plan, purpose, procedures and results of the tests, the extent of test coverage, the metric used for calculating extent, and a justification why the coverage is sufficient.

Discussion: The intent of the ITSEC authors was to require evidence of degree of test coverage by developers for individual functional units and for the trusted system as a whole. Because of the size and complex functionality of some trusted systems, extensive, let alone complete, test coverage is difficult to achieve. The developer and evaluator should know and be able to document what has been achieved through testing.

13. Fail to provide trusted distribution.

TCSEC: no requirement

ITSEC: §3.6.2.2.2 b) A procedure approved by the certification authority for this assurance level shall be followed, which guarantees the authenticity of the delivered TOE.

Discussion: This is an intentional requirement that extends the TCSEC.

14. Fail to provide checks against maintenance without agreement of the security administrator.

TCSEC:	no requirement
ITSEC:	§3.6.2.2.3 b) No maintenance shall be possible without the agreement of the administrator.
Discussion:	This is an intentional requirement that extends the TCSEC. Constraints in the trusted system are required so that the agreement of the administrator is assured before on-line maintenance is performed.

15. Fail to identify all security mechanisms and their interrelationships.

TCSEC:	no explicit requirement
ITSEC:	§3.6.1.1.4 b) It [the specification document] shall explain the realization of all security functions through all levels of design hierarchy, and identify all security mechanisms.
Discussion:	This requirement was intentionally included by the ITSEC authors. The requirement should be met by the philosophy of protection required by the TCSEC.

16. Fail to provide security functions that are adequately easy to use.

TCSEC:	no requirement
ITSEC:	§4.3.1 a) Under this aspect of assessment, the security functions and mechanisms of the TOE are assessed for their practicality of use in actual live operation.
Discussion:	This requirement was discussed in Brussels. It is likely to be reworded to make it more objective.

SUMMARY

As indicated by the previous sections, although they are similar, the F5/E5 and B3 requirements are not identical. Without explicit effort to meet additional requirements, a system targeted at one rating would not meet the other. An F5/E5 targeted system must meet additional or more constrained requirements on system structure, trusted path, labels on printed output, and object reuse. A B3 targeted system must take additional effort with system development practices, trusted distribution, and maintenance controls. Expressed another way, an F5/E5 system has more architectural freedom than B3 in achieving high assurance of confidentiality while a B3 system is less constrained in its development practices. For a B3 targeted system to achieve an F5/E5 rating, the following additional requirements must be met:

* Provide detailed design specifications with mappings between design levels.

* Use more elaborate configuration management tools.

* Use test coverage tools for unit testing.

* Develop trusted distribution procedures.

* Incorporate security administrator authorization for maintenance.

Analyzing the TCSEC to determine the impact of compliance with F5/E5 requirements upon a B3 system proved to be a very useful technique for determining the relationship between the ITSEC and the TCSEC. It caused the questions to become specific enough so that productive dialogue could take place with ITSEC authors to clarify the meaning of particular requirements. This resulted in better understanding of the document as a whole by those more familiar with the TCSEC and realization of the implications of ITSEC wording by its authors. In thirteen cases, specific intentional differences between the TCSEC and ITSEC were identified. In two instances, the participating authors felt that changes in the ITSEC were likely. Wording changes to clarify intent were deemed essential in nine cases. In two instances, the authors felt that clarification would occur in the manual for evaluation that is anticipated in the future.

Although the analysis of F5/E5 and B3 requirements does not provide a general comparison of the ITSEC with the TCSEC, it does serve to clarify some of the intended similarities and differences in the two documents. As such, the dialogue that ensued cannot but lead to the development of more precise and understandable criteria.

Since its first publication in 1983, there have been at least two broad types of criticism levied at the TCSEC. The first is that parts of it are ambiguous and imprecise. The TCSEC authors freely admit that there are inadequacies in the document. The ITSEC authors have tried to eliminate some of the difficulties of the TCSEC. Many of these points where the authors of the ITSEC have intentionally varied with the written or interpreted TCSEC lead to points where the ITSEC is stronger than the TCSEC. Being human, however, the ITSEC authors in their own writing have introduced ambiguity and imprecision which, ideally, will be clarified in future drafts. This paper has identified both points of intentional variation of the ITSEC from the TCSEC, and points of ambiguity in the current draft of the ITSEC.

A second major criticism of the TCSEC is that its binding of functionality and assurance into a single digraph class is too restrictive. The authors of the ITSEC have chosen to separate functionality and assurance completely, so that for example, evaluation of a high assurance-limited functionality trusted system becomes a possibility. Also, the authors of the ITSEC have decided to broaden its applicability by allowing the evaluation of trusted systems whose policy is other than confidentiality. These goals extend the applicability of the ITSEC beyond the range of trusted systems for which the TCSEC is appropriate. These goals also have the unfortunate side effect of allowing only minimal requirements to be posed for either functionality or assurance. To mandate specific mechanisms would be inappropriate since different policies may require different mechanisms. At low assurance levels, one might be willing to accept modest functionality, but one would want more stringent functionality requirements as the assurance level rises. Given the absolute separation of features from assurance, it was impossible for the ITSEC authors to impose such progressive requirements. While the ITSEC authors have addressed the excessive restrictiveness in the TCSEC, they have also become susceptible to the problems of generality.

The authors of the ITSEC used different premises and language then the TCSEC and thereby created an evaluation document that is close but not identical to the TCSEC. As has been identified in this analysis, some of the variations between the ITSEC and the TCSEC were intentional, while others were not. A goal of this analysis has been to clarify the differences so that as the authors of the ITSEC refine their criteria, only the intentional differences will remain. However, ignoring the predefined functionality classes (which are in any case only exemplary), the ITSEC represents a catalogue of evaluation criteria, whereas the TCSEC is a mixture of evaluation criteria and security requirements. The ITSEC does not (nor was it intended to) tell anyone what to build, only how to evaluate what has been built.

REFERENCES

[ITS90] Information Technology Security Evaluation Criteria, Draft version 1, May 1990.

[GISA89] German Information Security Agency, IT-Security Criteria, version 1, 1989.

[TCS85] National Computer Security Center, Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

AUDITING OF DISTRIBUTED SYSTEMS

D. Banning, G. Ellingwood, C. Franklin, C. Muckenhirn, D. Price
SPARTA, Inc.
7926 Jones Branch Drive
Suite 900
McLean, VA 22102

Security auditing systems are used to detect and assess unauthorized or abusive system usage. Until recently, security audits have been confined to a single computer system. Current work examines ways of extending auditing to include heterogeneous groups of computers (distributed systems). This paper examines the issues involved in auditing distributed systems, presents the framework for a Distributed Auditing System (DAS), and proposes a design for the audit reporting elements of the DAS.

INTRODUCTION

Security auditing for computer systems is the collection and analysis of computer system usage information used to ascertain the security posture of a computer system. Until recently, auditing has been performed only on a local basis, that is, information collected was logged on the system under audit. While this is a reasonable approach in an environment where there are few hosts that require auditing, as the number of hosts requiring audit increases, it becomes difficult to 1) examine the audit trails, 2) analyze the information and correlate events on one host to events on others, and 3) maintain consistent levels of audit collection. A further complication in large networks is the probable use of a variety of computer systems, each potentially having a different auditing mechanism, reporting syntax, and audit trail.

This paper presents an architecture for the collection of audit data in a distributed environment. One of the goals of this document is to relate the Distributed Audit System (DAS) architecture to the large body of work currently being done in the area of intrusion detection.

We are providing a method for presenting system-independent audit information and transportation of the information for analysis by a security officer or intrusion detection system at a central node in a distributed network. Our approach is expected to complement intrusion detection systems, not to compete with them.

Overview

The primary purpose of this paper is to describe a concept for auditing security-relevant events in a distributed environment. To accomplish this goal we defined the relevant audit issues, outlined the specific goals of auditing, and put our work in perspective with ongoing intrusion detection projects. These are briefly outlined here in order to accomplish the main focus as described above. An in-depth discussion of these issues can be found in the draft report delivered to Lawrence Livermore National Laboratory in September of 1990 and referenced in the bibliography.

The issues relevant to distributed auditing include: what data should be collected, how to transport audit data from a collection point to an analysis point, the system-independent audit data representation, the user interface and user invoked functions and the control of audit functions from a remote location. These are all issues that have been addressed in the concept and design of the DAS architecture as presented in Figure 1.

Other issues that are more appropriate for research by developers of intrusion detection systems include: data storage for subsequent retrieval and damage assessment, formulation of audit records into "security events" and anomaly detection from a set of events. What constitutes a good intrusion detection algorithm for network use is being addressed by projects such as Intrusion Detection Expert System (IDES), Haystack and the Network Security Monitor (NSM).

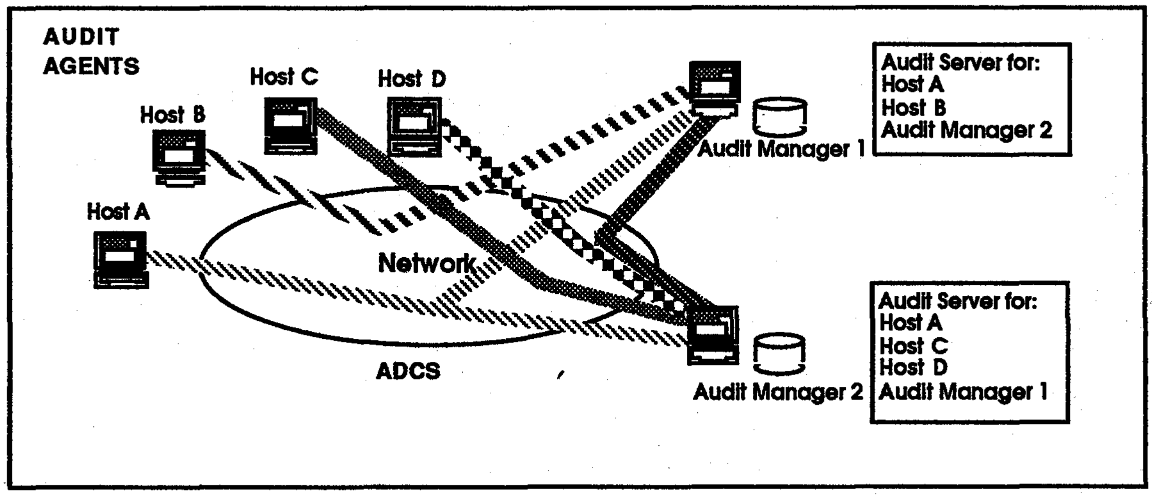

AUDIT AGENTS
Host A
Host B
Host C
Host D
Network
ADCS
Audit Manager 1
Audit Server for: Host A Host B Audit Manager 2
Audit Manager 2
Audit Server for: Host A Host C Host D Audit Manager 1

Figure 1 Distributed Auditing System Conceptual Architecture

GOALS OF AUDITING

This section briefly summarizes the goals of auditing and serves to establish the requirements for the design.

Security auditing is a broad function that can include the definition of security events, the creation of audit records, the real time analysis of these records for indications of anomalous activity, the archiving of these records for subsequent analysis, and the postmortem analysis of these archived records for various purposes.

From a security objectives standpoint, one of the more important goals of security auditing is that of providing for individual accountability, such that an individual knows with certainty that he is to be held accountable for his actions. This alone may serve as a major deterrent to abusive behavior.

Other related requirements for audit records are summarized below:

Intrusion Detection: the ability to detect suspicious activity through the use of user profiles

Real Time Monitoring: the ability to monitor activity on a system in order to detect unauthorized activity

Damage Assessment: the ability to determine what was compromised

Attack Reconstruction: the ability to understand how an attack was carried out (i.e., in order to design effective countermeasures to guard against future attacks of the same type)

Damage Recovery: the ability to recover from whatever damage may have occurred

Each application may require an additional set of information that the security auditing system should collect. A good auditing system should address all applications that have a requirement for audit records.

The approach has been to define an overall security auditing architecture with functions and mechanisms for the collection and management of audit-related data, and allowing for the future refinement of these mechanisms to serve advanced requirements such as computer analysis.

DAS CONCEPT EVOLUTION

The history of the DAS began in 1988 with the initial concept of a "virtual audit trail". It has evolved into the current definition of a set of network management protocols to control the collection of audit data

Initial Concept

The original concept defined a standard representation of a canonical audit trail that could be used by the current audit analysis tools. Standard form records collected on multiple machines could be analyzed by a single security monitor for an entire network of systems. This concept evolved into the notion of a "virtual audit trail" and a related set of protocols for transporting data in a common format. In considering complex situations where multiple "audit messages" are needed to compose a single "network virtual audit message", a method was considered where the "translation" would be performed at the application level and the presentation protocol would be used for local data-representation translation.

Different types of transport protocols such as, TCP, UDP and VMTP were considered with respect to the selection of transport reliability, duration of calls and use of network resources. The main difference between the different transport protocols is in how they move data (e.g., as independent blocks of data or as a continuous stream of bytes) and how reliability is achieved.

Evolution of the Concept

An architecture for distributed auditing developed as mechanisms were described for collecting data from multiple host systems in a network for a multitude of purposes (e.g., real time intrusion detection or after the fact analysis resulting in damage assessment). The architecture provides a framework for a set of application/transport level protocols for transmission of auditing data, and a management protocol for controlling the local host (e.g., setting thresholds and synchronizing clocks) from a remote location.

A top level outline of a DAS was developed and documented in a report delivered to LLNL in September 1989. Later, the notion of an auditing protocol was extended to address both the transmission of data from the Audit Agent (AA) to the Audit Manager (AM) and the control the operation of the AA from the AM. The names of these components have evolved to allow association with terms more commonly used in network management.

The belief is that the AM can send commands to the AA (via the Audit Data Communication Service (ADCS))to increase granularity of monitoring, to audit specific users in detail, to audit accesses to specific files, to audit specific system calls, log all traffic to/from a specific node/terminal, take a snapshot core image, etc. Thus the security officer, sitting at a workstation connected to the AM, can control the auditing throughout the entire network and can respond quickly to newly discovered attacks (e.g., as those reported on the networks by CERT and LLNL's CIAC).

The machine that supports the AM can also have a back-end connection to a system that interprets the audit information for real time detection of anomalous events. An extension is to allow such a system to signal the AM to increase the fidelity of monitoring, etc., much as a human security officer would respond to detected anomalous events. The concept can be further extended to the idea of multiple AMs, where each community of interest can have its own AM, allowing logical subnets for which each AM collects audit data.

Network Management as a Model for Auditing

SPARTA's Networking Research group is heavily involved in the design and development of network management protocols. Struck by the similarity between collecting audit data and collecting performance data, they suggested that we examine the work in the network R&D community that is leading to the definition of network management protocols providing mechanisms for collecting data from various nodes in a network. They observed that, since the mechanisms for controlling the collection process and for reporting it to a central site are similar, the same protocols might be used for both purposes.

A review of the evolving specifications for the upcoming Common Management Information Protocol (CMIP), the related CMI Service (CMIS) specification, and the Management Information Base (MIB) which defines the data elements showed the similarity between collecting data in a network for network management and collecting security-relevant data elements.

A copy of CMOT (CMIP running over TCP) was obtained from the University of Wisconsin and was evaluated on the company's internal LAN to determine whether it could be easily extended to collect the network security data elements.

We concluded that network management protocols provide a good model for our DAS design and the network auditing architecture and design presented in this document is based on the premise of extending the network management protocols currently being defined to incorporate provisions for the collection of security events.

The DAS concept now includes the following:

1. An application for collecting data and transforming it to a network virtual representation. This requires a format and semantic meaning for audit records
2. A transport protocol for transmitting the audit records.
3. A management protocol for dispatching commands from the central site to the remote node that requires a format and semantic meaning for the commands (i.e., to instruct the remote node how to behave upon receipt of each command).

AUDITING AS A NETWORK MANAGEMENT FUNCTION

Network management protocols provide a mechanism for transmitting network performance information from remote nodes to a central collection point. As mentioned earlier, the collection and reporting process for performance data and audit data are very similar. Therefore, the network management protocols can serve as a "model" for collecting, reporting, and transmitting audit information in a distributed network. Below is a brief description of network management protocols and their applicability to audit functions.

Introduction to Network Management

Network management is accomplished by managers at local management stations and agents at remote managed nodes exchanging monitoring and control information via protocols and shared conceptual schema about a network and its components. The shared conceptual schema mentioned above is a priori knowledge about "managed objects" concerning which information is to be exchanged. Managed objects are abstractions of system and networking resources (e.g., a protocol entity, an IP routing table, or in this case, auditing resources) that are subject to management. Managed objects have attributes, operations, and notifications that are visible to managers. The internal functioning of the managed object is not visible to the manager. Currently, an agent is responsible for conversions between a managed system's internal format of managed objects and the external format of managed objects (i.e., the form expected by the manager).

Using management services and protocols, a manager can direct an agent to perform an operation on a managed object for which it is responsible. Such operations might be to return certain values associated with a managed object (i.e., get a variable), to change certain values associated with a managed object (i.e., set a variable), or perform an action, such as self-test, on a managed object. In addition, the agent may also forward to the manager notifications generated asynchronously by managed objects (e.g., send updates periodically).

Network Management Architecture

The network management architecture described here consists of a Management Information Base (MIB) containing a list of managed objects, the International Organization for Standardization (ISO) Common Management Information Services (CMIS)/Common Management Information Protocol (CMIP) Manager and Agents. The Managers and Agents exchange information based on the managed object definitions contained in the MIB, and the ISO network management protocols that facilitate the exchange of this information.

Management Information Base (MIB)

A MIB is a list of managed objects, described in external format, which are considered useful for a particular application. The Internet MIB contains managed objects that are read-only (since current management protocols are not sufficiently secure to exert control, as would be the case with writable objects), and help a manager determine the status of the network elements. Using the Internet MIB as a model, it should be possible to develop an audit MIB.

CMIS/CMIP Manager and Agents

The Common Management Information Services (CMIS) are provided by the Common Management Information Service Element (CMISE). The Common Management Information Protocol (CMIP) supports these services. An invoking CMISE-service-user, or "manager", may invoke a management operation. A performing CMISE-service-user, or "agent", is the process that performs a management operation invoked by a "manager." A CMIS/CMIP manager and agent applications could use adaptations of the ISO Common Management Information Service Element (CMISE) to exchange information and commands for the purpose of auditing.

CMISE provides facilities for a managed "agent" to send multiple linked responses to a manager. An audit agent could use this type of service to send detailed information to an audit manager.

CMISE also provides to managers the ability to "multicast" operations to be performed on a group of managed objects. Through CMISE services, a manager can perform a single operation on a group of managed objects. A distributed audit mechanism could use such a service to assist in responding interactively to network attacks.

4.3 Uses of CMIP in Distributed Auditing

CMIP offers a mechanism to transmit information between agents and managers in a distributed network. The components of the auditing system could use the network communication services offered by CMIP.

AMs located on remote network nodes can send messages to audit applications located on many different local nodes. Audit applications would use the same services to send audit information to the AMs. The advantage of using CMIP for such communication is that a rudimentary mechanism already exists through the CMISE services.

To implement a distributed audit capability using the CMIP protocol for communication, the CMIP protocol would have to be extended. Additions to CMIP would include definition of message types to transmit between manager and agent and specification of what information is expected of both the manager and the agent.

DISTRIBUTED AUDIT SYSTEM DESIGN

The design of the DAS consists of 4 major components, Virtual Audit Trail (VAT), Audit Agent (AA), Audit Manager (AM), and Audit Data Communication Service (ADCS) as depicted in Figure 2. The functions performed by each of these components are discussed below.

Security of the DAS is critical to a successful implementation of audit services. Without the implementation of security principles, a DAS may be attacked and rendered useless in either detecting an attack on other computing resources or in assessing the cause and extent of any resulting loss.

The DAS Architecture incorporates three security principles: access control, data integrity, and assured delivery of messages. In light of this: 1) only specifically authorized individuals (usually a security officer) may change the selection of audited events on a system or cause the audit reporting mechanism/process to stop; 2) audit reports must not be modified while in storage or in transit to storage (over the network); and 3) audit reports that are generated and transmitted to a manager must be received.

Virtual Audit Trail (VAT)

The VAT is formulated from audit information sent from the AA to the AM. A virtual audit record is distinct from what is recorded on a particular host. It is O/S independent and reflects security relevant events and must be inclusive enough to fulfill any of the goals outlined in Section 2. The virtual audit record is unrestricted by what the local site security policy defines as security relevant.

To determine what constitutes such an audit record we should examine several areas:

1) look at the auditing done by particular O/Ss, determine the security relevant events and include these in the virtual audit record,

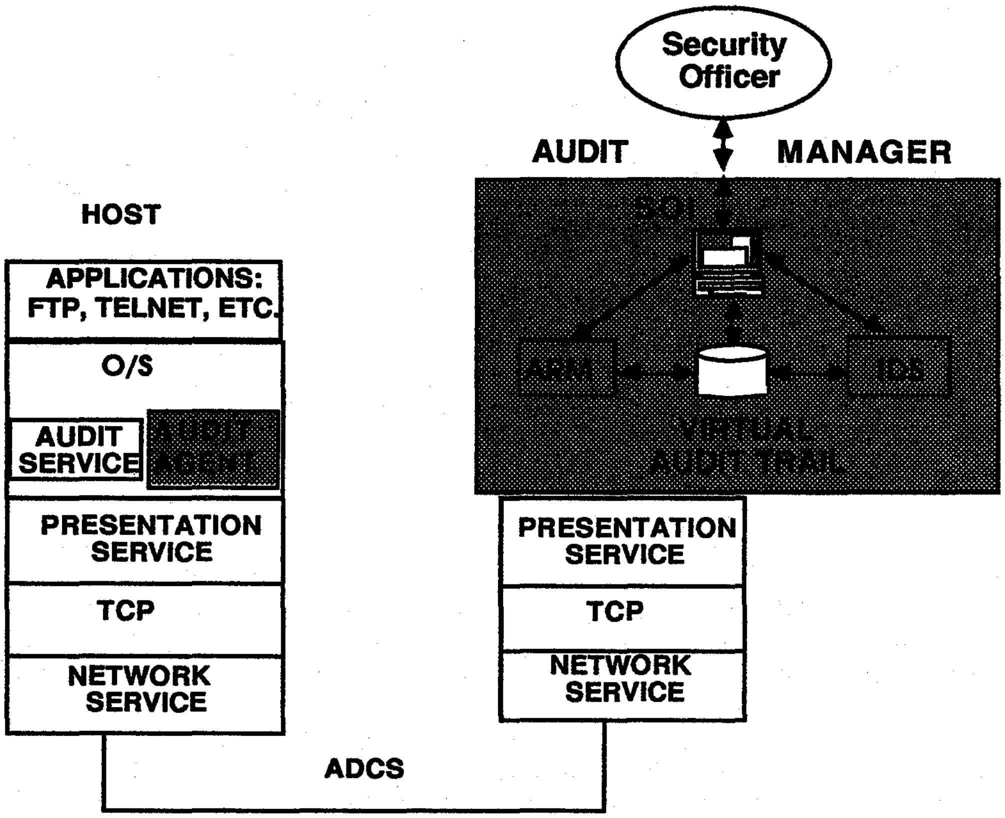


Figure 2. Distributed Audit System Design

2) look at the current audit analysis tools and catalogue what events are needed by each of them for their particular analysis, and
3) look at what events have triggered discovery of incidents in a real situation (e.g., Cliff Stoll's incident, etc.).

Using the above information, a set of record types that represent different types of events can be defined. For each type of record, the variables that define that event are determined These audit records constitute the VAT at the AM.

Once the contents of the audit records have been defined, a MIB of audited elements is specified for use with the network management protocols. An audit MIB contains managed objects considered essential for auditing.

Audit Agent (AA)

The AA consists of a process running on each network host and has three principle functions: selecting audit events for forwarding to the AM, translating host-specific information into a "virtual" format, and responding to commands from the AM.

The AA sends selected audit information from the host to the remote AM. In a distributed network, each host would have an AA and would report to a number of AMs. The AA examines the audit records generated by the host's O/S and determines what information to forward by examining an audit table.

Forwarding Audit Events

This audit table, depicted in Figure 3 is maintained and updated in response to commands from the AM. The use of the audit table allows each site to send audit reports based upon the site's individual security policy.

The audit table tells the AA which events to send as event reports, which to send as event summaries, and which to ignore. An event report is a detailed record containing information such as the userid, command invoked, network address, and any related fields specific to a particular event. An event summary reports the frequency and number of a particular event per some unit time. The event summary could be useful in a real-time situation where limited specific information is needed quickly (e.g., when an intrusion is suspected and more information is needed).

The audit table is read by the AA upon initialization. The audit table has the structure of username, event, report, and summary. The username field indicates which user's activities are to be audited. The event field indicates what event to audit. The report field is a boolean value that indicates if an event report is to be sent to a Audit Manager. The summary field is also a boolean value that indicates if an event summary is to be sent to the Audit Manager. Usually, the event, and report fields are mutually exclusive, i.e., you either send an event report or an event summary but not both. Finally the AM field indicates to which audit manager(s) this event should be reported.

Translating Host-Specific Information

The AA will use a language tailored to each O/S to perform translation of host-specific information to a "virtual" format. The language will consist of a set of verbs and nouns which express all the audit events to be used in the DAS. It is expected that this language will be extensive in order to express all the required information with the desired level of granularity.

Using this approach, logon reports could be as simple as "Joe logged on at 1:30" or as complex as "Sam, aliased to Joe, logged onto host Euler from host Kepler, whose internet address is 192.48.111.1, via the Internet gateway 192.5.8.1, on 26 June 1989 at 1:30 pm." Each of these reports is optimal for the information they contain. Each report relays all the information available from their respective O/Ss without loss or overhead.

Responding to Commands from the AM

The audit information collected by a particular AA is determined by local security policy. What subset of this information is sent to the AM is predetermined by the AA's audit table. Though this information would be periodically updated by the AM, it would be useful to have the ability to request further information from the AA.

The DAS provides the AM the capability of controlling the operation of the AA through a series of commands sent via the ADCS. These commands allow the AM to request increased granularity of audit information on specific: users, files, system calls, resources, and node/terminal traffic. Upon receipt of the commands the AA processes them, performs the necessary action and provides a response. If the necessary action cannot be performed (e.g., user has logged off and no further information can be obtained), a response indicating the inability to complete the task is formulated.

Audit Manager (AM)

The AM consists of three components: Audit Record Manager (ARM), Security Officer Interface (SOI) and the Intrusion Detection System (IDS). The AM acts as a centralized control center for audit information transmitted from distributed hosts. The three components of the AM work closely together to provide these services: collection/correlation of audit information, interpretation of audit information, and notification of the AA to take further action. Figure 4 shows the logical interrelationships between the AM components.

Audit Record Manager (ARM)

Upon receiving audit records from the AA, the ARM updates the audit database with the new information. Some maintenance functions are provided automatically (e.g., archiving and deletion of duplicate entries). Other functions are provided through a set of security officer queries entered

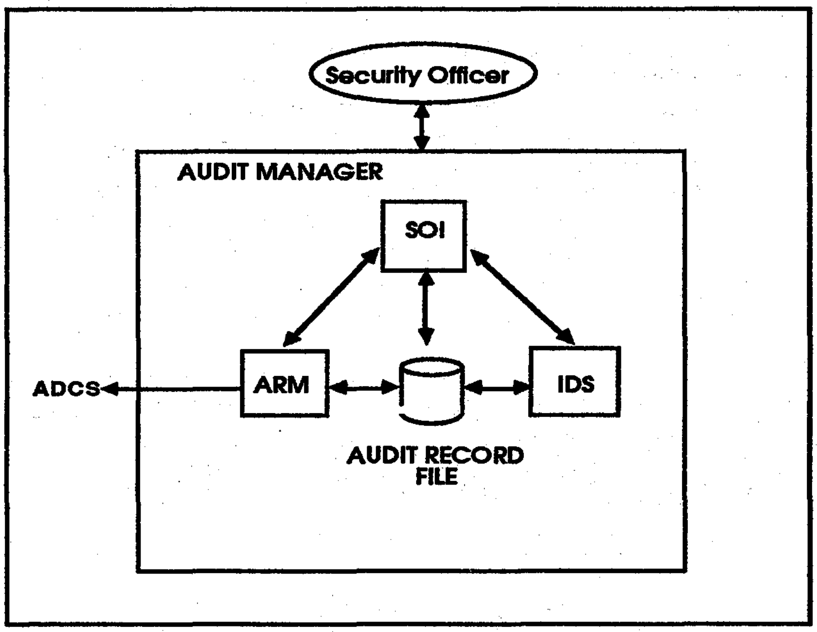


Figure 4. Audit Manager Architecture

through the SOI (e.g.,) deletion by record, correlation of audit entries and record retrieval. The ARM initiates transmission of audit table updates at specified time intervals. The ARM also has the capability of sending audit table updates upon instruction from the SOI.

A primary purpose of the audit database is to provide the necessary information for the IDS for identifying suspicious activity. Querying of the audit database can be done through the IDS or via the SOI and controlled by the ARM.

The ARM also provides correlation of incoming audit information from different hosts. This correlated information is then given to the IDS for analysis. Correlation of information is important for those networks where the same user utilizes different hosts such that a complete set of audit information can be given to the IDS for analysis.

<u>Security Officer Interface (SOI)</u>

The SOI provides an information display and command processing capability. The SOI display will use a window structure to provide graphical display of detected anomalies, security of the network and status of AM functions. A command capability will be provided for issuing commands to the AA for additional audit information. A menu of frequently used commands will be provided as well as a command line option.

<u>Intrusion Detection System (IDS)</u>

The IDS to be used with the Distributed Audit System is not specified in this design, but treated as a "black box" that uses the audit records maintained by the ARM to detect suspicious activity. The IDS used for this function can be any of the current systems available. The configuration or function of the IDS is independent of its use for this DAS design.

The DAS will provide audit records to the IDS for analysis of user activity. The security officer will then be able to send a command to the AA requesting an additional granularity of information on a particular user. For example, if the AM receives an event summary that user Joe has used the telnet command 50 times in the past hour and this activity is outside of Joe's user profile (according to the IDS), the AM can send a message to the AA asking to see all the commands issued by Joe. When the AA receives this request, it would modify the audit table to reflect the request to monitor Joe more closely.

However, if an IDS is used that does not perform real-time monitoring, the additional information available will be limited to that already in the AA audit trail since the user will most likely not be active.

Audit Data Communication Service (ADCS)

The ADCS provides the necessary communication services for transporting messages between AA and AM. To enable an AM to control the functions of an AA, services currently defined by CMIS could be adapted for use in the ADCS. Using the network management services provided by the ADCS, the AM could request the AA to provide additional audit records on a particular user or event, change the events being audited, set/reset audit thresholds, and provide event reporting at specified intervals.

The automatic reporting of audit events to the AM from the AA could be accomplished using the M-EVENT-REPORT service which is invoked by the AA at specified intervals.

Using the CMIS management services and CMIP protocol, the manager can direct the agent to perform an operation on a managed object for which it is responsible. The following services would be invoked by the AM to make requests of an AA:

M-GET: Used to request additional audit records from the AA for increased granularity from existing audit records.

M-SET: Used for setting/resetting AA audit thresholds from the AM.

M-ACTION: Used to increase collection of data by modifying an existing parameter (e.g., change the system files to be audited).

M-CREATE: Used to request an AA to audit new events for a particular user.

M-DELETE: Used to request AA delete audit records, audit events or an audited user due to changes in operation.

The ADCS must also provide the security services of data confidentiality, data integrity during transmission and assured service of messages.

OTHER ISSUES AND FUTURE DIRECTIONS

As indicated in the overview section, many issues were considered in the DAS design and not all of them can be thoroughly discussed here. To fully define the DAS design, it is necessary to resolve some additional audit issues that are currently being researched. These include, but are not limited to the security and technology issues outlined briefly below.

Security issues related to the building of a DAS include:

- Assurance - both in the case of being assured that the AA is performing as it should and in the case of being assured that the AM is secure from penetration;
- Transmission security - the information flow from the AAs to the AMs and vice versa must be secure; and
- Network Management Protocol Security - while work is ongoing in this area, the idea is still fairly new.

Technology issues facing the successful implementation of a DAS include:

- Commercial Marketability;
- Anomaly Detection Capability - the testing of; and
- Time Stamping - addressing the delays related with heterogeneous hosts.

All of these issues can be addressed via prototyping, which is the next step in the process.

BIBLIOGRAPHY

[1] DOD5200.28-STD, DOD Trusted Computer System Evaluation Criteria (TCSEC), December 1985.

[2] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 9595, CMIS, 6 December 1989.

[3] International Organization for Standardization (ISO)/International Electrotechnica[Commission (IEC) 9596, CMIP, 6 December 1989.

[4] Internet Request For Comments (RFC) 1156, MIB, May 1990.

[5] Internet RFC 1155, Structure and Identification of Management Information for TCP/IP-based Internet (SMI), May 1990.

[6] Internet RFC 1095, The Common Management Information Services and Protocol over TCP/IP (CMOT), April 1989.

[7] NCSC-TG-001, Version 2, A Guide to Understanding Audit in Trusted Systems, 1 June 1988.

[8] NCSC-TG-005, Version 1, Trusted Network Interpretation, 31 July 1987.

[9] SPD1003.6, Draft 2, Security Interface for the Portable Operating System Interface for Computer Environments, 4 June 1989.

[10] Auditing of Distributed Systems (Draft), SPARTA, Inc., 28 September 1990

[11] Survey of Sudit Trails and Audit Analysis Tools, SPARTA, Inc., 3 March 1989

BUILDING A MULTI-LEVEL APPLICATION ON AN UNTRUSTED DBMS IN A UNIX SYSTEM V/MLS ENVIRONMENT - A PROJECT'S EXPERIENCE

David S. Crawford
Directorate of Security Operations
Department of National Defence
National Defence Headquarters
101 Colonel By Drive
Ottawa, Ontario, Canada K1A 0K2
(613) 993-6775

ABSTRACT The procurement option of using an untrusted DBMS on a TCB where both trusted system and DBMS functionality is required is briefly discussed in Appendix B of the Trusted Database Interpretation. This paper discusses an approach proposed for a Canadian Department of National Defence project to design and implement several multilevel multiuser DBMS-based applications using an untrusted DBMS on a B1 UNIX® TCB, and the design and operational constraints imposed by this solution.

PART I - INTRODUCTION

1.0 BACKGROUND

Adequate segregation of sensitive information has historically been a serious impediment to the provision of Information Technology services in support of defence activities. One Canadian Department of National Defence project had concerns about the ability to provide a secure environment for applications and data on Base level computer systems due to presence of both UNCLASSIFIED and CONFIDENTIAL information. Data analysis had determined that information processed on these applications was, in certain instances, classified in isolation and in aggregation. In addition, the number of sites involved resulted in significant cost implications if all equipment at all sites was required to meet TEMPEST standards, since current Canadian standards require TEMPEST protection for any classified processing.

These concerns led to the project to plan to operate in a Controlled (restricted form of multilevel) Security Mode of Operation and the statement of a requirement for a B1 Trusted Computer Base, which was subsequently specified as AT&T UNIX® System V/MLS (SV/MLS)[1]. By specifying a B Division TCB, the project intended to address confidentiality concerns and to minimize the number of TEMPEST equipment required, since device labelling could be used to restrict classified processing to only the limited number of TEMPEST devices attached to the TEMPEST host computer. Other integrity concerns would be addressed by traditional software engineering practices.

The other early concern for the project was establishing the application software environment. Procurement of a

1 UNIX is a registered trade mark of AT&T

DBMS and 4GL environment was initiated and resulted in the procurement of ZIM®[2], a DBMS and 4GL from Sterling Software, for this project prior to determination of the TCB requirements.

The framework of untrusted DBMS and secure UNIX was established without considering whether or not the DBMS could be effectively used on a multilevel operating system and how the DBMS based applications could be designed. It now remained to determine how to design and implement DBMS based applications that would meet security requirements without violating the TCB.

This paper discusses major design issues necessary to build multilevel applications within the project constraints and additional considerations employed to provide additional protection.

2.0 PROJECT FRAMEWORK AND CONSTRAINTS

The nature of a multilevel application is that it more closely models an actual defence-related environment, where information exists at various levels of sensitivity. More traditional data processing approaches, such as operating separate systems for various levels of sensitive information or treating all information at the highest level of sensitivity held, are expensive both in terms of capital procurement costs and administrative overhead. From the project perspective, operating with UNCLASSIFIED and CONFIDENTIAL information would require all project equipment to meet TEMPEST requirements unless an acceptable multilevel solution could be implemented.

The project, as part of the requirements definition, had conducted extensive data modelling. Analysis of the information model from a security perspective established that any tuple, in isolation, was UNCLASSIFIED. However, specific tables were identified that were, in the aggregate, CONFIDENTIAL. These tables were relatively static and managed in isolation by a central authority.

In addition, specific joined tables, in the aggregate, were CONFIDENTIAL. Project personnel were able to identify specific views, application screens and reports that contained classified information.

One area of considerable concern dealing with aggregation concerned the quantity at which the aggregate became classified. The classic example on the project was the aggregation of persons, where an individual tuple was UNCLASSIFIED and all persons belonging to a unit reveal operational capability and thus was CONFIDENTIAL. The project solution was to establish an overly restrictive de-facto business rule that any set containing more than one tuple was classified.

Project applications would be developed and maintained by a central authority. Each application would be released to sites as a turnkey application or subsequently as an update to an application. No capability to modify the applications was to be provided to the field.

PART II - DESIGN FRAMEWORK

The specific conditions within the project and the features available in the TCB and the DBMS led to the formulation of two general problems and the associated approaches in building multilevel applications that relied on TCB controls and the identification of additional controls that would compensate for acknowledged weaknesses.

[2] ZIM is a registered trade mark of Sterling Software

3.0 SECURING DBMS TABLES - A FILE BASED APPROACH

The first general problem was controlling access to any data within a given DBMS table and was based on the existence of tables that contained CONFIDENTIAL information. The general approach taken to address this problem was based on the use of the TCB mandatory access controls to control access to DBMS tables. This approach involved the labelling of the O/S files containing the DBMS tables according to the highest level of sensitivity of the DBMS table, thus controlling access to data through TCB controls.

The extensive data analysis on the project supported this approach since it was readily apparent that tuples within tables could be assumed to be of a uniform sensitivity and table level sensitivity labelling would be sufficient. This approach would have not be appropriate had tuples within tables been required to reflect differing levels of sensitivity.

This approach was technically possible in the target environment since the ZIM DBMS managed each table as a separate O/S file. The DBMS only opened those tables required and opened tables as READ-ONLY unless the table was being updated. Errors in opening tables for WRITE access, such as are caused by opening files labelled at a lower level, resulted in the SELECT operation returning a null set and a warning message issued by the DBMS.

One concern with this approach is that it may impose significant restrictions on functionality if update activities spanning classification levels are necessary. In the case of this project, most tables were UNCLASSIFIED. The few CONFIDENTIAL tables were relatively static, were not closely linked to its related UNCLASSIFIED tables and could be maintained independent of the UNCLASSIFIED tables.

4.0 SECURING DBMS VIEWS - A PROCESS BASED APPROACH

The second general problem was controlling access to CONFIDENTIAL views of data that was UNCLASSIFIED in isolation. The approach to address this problem was made somewhat obtuse since the ZIM DBMS did not directly support a view mechanism. However the view mechanism was represented through each Selection and Projection operation in each ZIM program.

A means to address this problem was needed. A view was represented as the retrieval statement, such as a SELECT statement, within a program. Each ZIM program existed as an O/S file and the DBMS required READ access to the file in order to execute the program. By labelling each program with a sensitivity label corresponding to the highest level of sensitivity of the views or aggregations being manipulated, the TCB was employed to control access to views and aggregations. Access would be based on the sensitivity label of the user's process that invoked the program, hence the term "process based" control. This meant that users operating at a level dominating the program label could execute the program whereas users operating at a lower level would be unable to execute the program. Based on this approach, it was accepted that labelling the means of producing views or aggregations would represent a comparable functionality to labelling views.

This approach was supported by earlier work on the project to define screen and report formats and contents. This work had included review for security relevant issues, such as display of classified information.

An additional refinement to this approach sought to employ mandatory controls to enforce some integrity issues by using confidentiality labels as *de-facto* integrity labels. The labelling scheme was modified so that application programs would be labelled at a <level - 1> in order to isolate the programs from the user processes. In the project example, UNCLASSIFIED was established as level 30 and CONFIDENTIAL was level 180. Level 29 was created for UNCLASSIFIED programs and level 179 was created for CONFIDENTIAL programs.

Level Name	Level Number (numeric level)	Suffix	Prompt	Hardcopy
Secret	210	(S)	SECRET	SECRET
Application S	209	(Appl S)	Appl (S)	Appl (S)
Confidential	180	(C)	CONFIDENTIAL	CONFIDENTIAL
Application C	179	(Appl C)	APPL (C)	APPL (C)
Protected A	60	(PA)	PROTECTED A	PROTECTED A
Application PA	59	(Appl PA)	APPL (PA)	APPL (PA)
Unclassified	30	(U)	UNCLASSIFIED	UNCLASSIFIED
Application U	29	(Appl U)	APPL (U)	APPL (U)
System	0	(TCB)	SYSTEM	SYSTEM

Figure 1: Project Labelling Hierarchy

5.0 ADDITIONAL CONTROLS

In considering a process based approach to managing access to data, the software engineer must consider both controlled access and uncontrolled access to sensitive information. Controlled access to information is the access that a user has through the application functionality and is a direct result of system design and implementation. This type of access is defined in terms of application screens, reports and query facilities. Uncontrolled access is the access that a user may have if free to specify how and what to retrieve. This type of access is typified by the use of ad-hoc query languages or through the use of other software, such as system utilities. In addition, access to information also includes device level considerations as there must be mechanisms in place to ensure that classified information is routed to the appropriate devices and labelled appropriately.

Uncontrolled access to information poses the most immediate threat in the use of a process based approach to building a multilevel application. This is primarily due to the potential for uncontrolled aggregations permitted through ad-hoc query facilities. The ability of a user to extemporaneously, repetitively and interactively define and retrieve any possible combination or permutation of data existing on a system poses a horrendous burden of proof on the software engineer that all possible data retrievals will be at the same level of sensitivity as the base data. In the case of this project, it was already known that some aggregations of data were CONFIDENTIAL, even though these combinations are based on data which is UNCLASSIFIED in isolation. This implies that access to the ad-hoc query capability, if permitted, be restricted to known users with the appropriate clearances and permissions, to users operating at the appropriate security level and to TEMPEST devices, if classified aggregates are possible.

There are two aspects to the ad-hoc query threat. There is the possible surreptitious access to underlying query capability. This is represented by users who circumvent controls and use software they are otherwise unauthorized to use. The second and more plausible threat is that of legitimate access to an ad-hoc query capability. Since the designer cannot control what the end user specifies as retrieval criteria, there are legitimate concerns that users could intentionally or inadvertently retrieve sensitive aggregations while operating at inappropriate security levels, while operating without the appropriate clearances or while using inappropriate (non-TEMPEST) devices.

The requirement to permit ad-hoc query can be very real for the applications designer as it will add considerable functionality in terms of addressing unforeseen information requirements and may significantly reduce the number of report generators required to be developed. The problem of controlling the contents of a query can only realistically be addressed by application software to pre-screen each query, a daunting software development task. However if the designer cannot control the contents of the query, he can control access to the means to query by denying unauthorized users and devices access to the query engine.

The problem of actual or potential access to an ad-hoc query facility, if sensitive aggregations can exist, will require that all applications be executed on a runtime engine. This requirement is necessary since it is imperative to guarantee that inappropriate end users or devices cannot, either intentionally or inadvertently, access any ad-hoc query facility. This assurance cannot be provided by application software. However, the operating system, since it is a TCB, can provide this assurance.

The use of the TCB mechanism of mandatory access controls can be employed to permit selective access to the ad-hoc query facility. The design of the ZIM engine assisted in that it did not use a client/server architecture but was separately invoked by each process executing the file. It was therefore possible to restrict access to the ad-hoc query facility by labelling the query runtime engine (executable file) at the CONFIDENTIAL level, which will make it inaccessible to users operating at levels lower than CONFIDENTIAL. Provided that the UNCLASSIFIED components of an application use the ZIM runtime engine, it was then possible to provide the functionality of ad-hoc query for users operating at a CONFIDENTIAL level without compromising access to the means to create classified aggregations.

One problem with this approach was that there were several smaller sites where more than one application would be hosted on the same CPU. In order to address this problem, the use of mandatory access controls in the form of application specific categories was established to enforce mandatory need to know separation of incompatible communities of interest. Since an application that does not hold information which is sensitive in the aggregate should not have restrictions placed on access to an ad-hoc query capability, such applications co-resident with a second application holding information which is sensitive in the aggregate could employ mandatory access controls in the form of application specific categories to differentiate between applications. Separate copies of the query runtime engine, each labelled with the appropriate security level and application specific category, would exist on the system. Users belonging to the second application would not be able to access the query runtime engine labelled for the first application and would, provided that they are restricted to a runtime engine, be unable to gain access to an ad-hoc query capability.

The requirement to ensure all classified or potentially classified information is routed to TEMPEST devices can be effectively addressed if access to classified aggregations is restricted to users operating at an appropriate classified level. SV/MLS supported device labelling whereby minimum and maximum clearance levels are assigned to devices, such as terminals and printers, by the system or security administrator. Labelling all non-TEMPEST terminals and printers with a maximum level of UNCLASSIFIED and all TEMPEST devices with the maximum level of CONFIDENTIAL provided assurances that potentially sensitive information could only be displayed or printed at appropriate devices.

The need to label screens with appropriate sensitivity labels was identified as a requirement. Label processing on SV/MLS required privileged system calls. The DBMS had a feature that enabled reading and writing to UNIX pipes. This feature enabled a very small, simple piece of untrusted application code to be developed to read the *stdout* output from the SV/MLS ***labels -u*** command, a trusted program that was part of the TCB, and extract the sensitivity label of the process knowing that the label was correct. Actual screen labelling was not trusted but was to be considered part of the normal software activity and subject to independent verification and validation.

Additional controls that were felt to be required included removing all access to the operating system interface ("prompt"). All project applications would deny access to the O/S prompt through the development of application specific menus installed as default shell. In addition, the removal, imposition of restrictive labelling or restricted file permissions on O/S shells and utilities would be carefully considered prior to system implementation. This was not seen as being detrimental to the project since systems were to be considered turnkey application specific systems and not general purpose ADP equipment.

PART III - DBMS CONSIDERATIONS

A number of DBMS related issues were encountered in building a prototype multilevel application on ZIM under SV/MLS. There are a number of areas where the SV/MLS environment impacts the use of ZIM and the application design. These areas do not, in general, represent problems which cannot be addressed but have approaches that solve, avoid or work around the difficulties.

One SV/MLS feature that proved key to the ability to implement an untrusted DBMS on a secure UNIX platform was secured, or multilevel directories. This feature was developed to address the problem caused by the widespread use of common directories with global read/write access, such as */tmp*, in UNIX. This feature enables the user to reference the same directory from more than one level, while the operating system transparently redirects the user into an appropriate subdirectory for the user's privilege (security) level. Untrusted subjects can reference the same directory and be transparently redirected to a directory which is appropriate for the subject's privilege. Trusted subjects, on the other hand, are not subject to this redirection and the entire directory structure is both visible and accessible [1].

6.0 DBMS WORK FILES

ZIM uses several working files (*zimsetd, zimsett*) on a per user session basis and which require READ/WRITE access. The SV/MLS environment impacts this DBMS requirement in that each user will require read/write access to their ZIM working files at all times and ZIM creates and maintains these files in the defined **work** directory. The immediate problem is that these files will exist at the same security level as the process creating them, which poses a problem in the case of an application requiring two or more security levels.

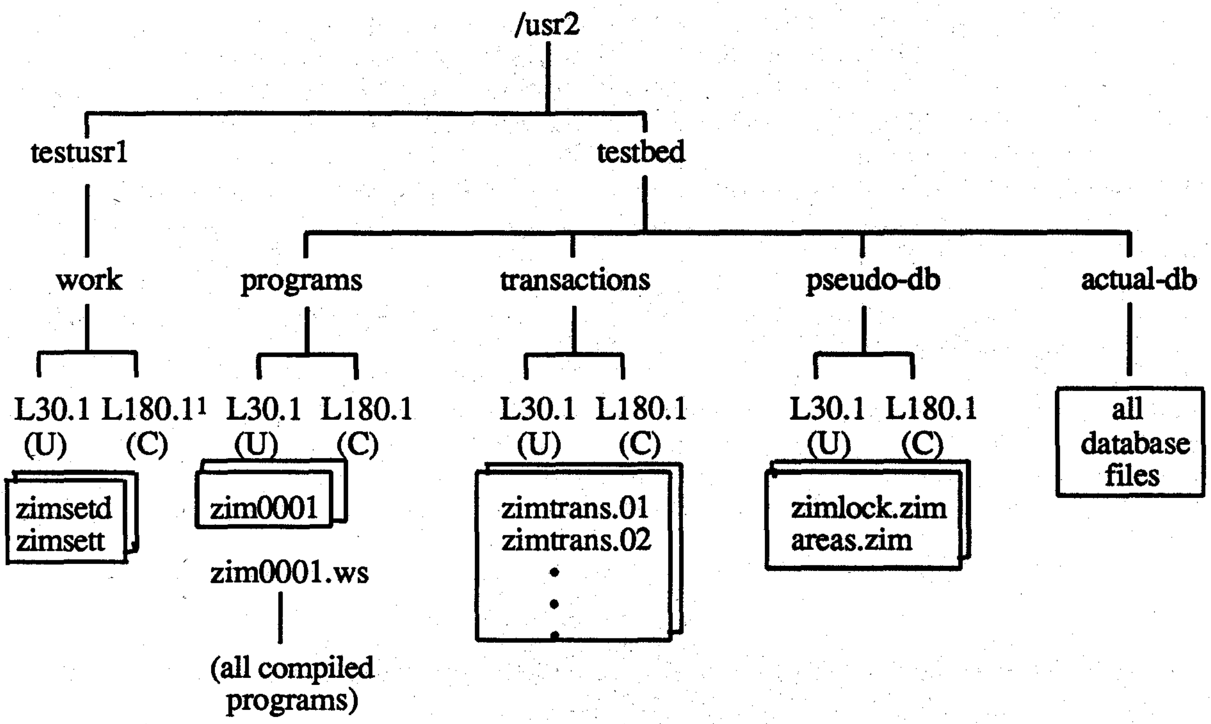
It was possible to set the working directory to a specific directory through the "***work path <pathname>***" entry in the ZIM configuration file (*config.zim*). By creating the working directory as a multilevel directory, a user account could be set up that would permit multilevel use of the DBMS.

7.0 DATA DICTIONARY

The ZIM data dictionary points to the location of all interpreted ZIM program files and all compiled ZIM programs are located in the *zim0001.ws* directory. ZIM, as of Release 3.03, required read/write access to the ZIM data dictionary (*zim0001*). This was subsequently modified to READ/ONLY access as a result of the DBMS port to the UNIX System V/MLS platform. The DBMS data dictionary did not now pose a problem in a SV/MLS environment since this file was now accessible to any process existing at a dominating level.

8.0 TRANSACTION FILES

ZIM used, in support of database journalling, a pool of transaction files that are used by all users of a database. Since these files are reused by all users of the database, the label associated with all journal files must be identical with that of all users. If users operating at a range of more than one label access the database, the DBMS will fail to function since journal files may not have identical labels. The location of zim transaction files (*zimtrans.n*) posed a possible problem as these files normally reside in the database directory. However, the transaction files can be redirected to another directory through the "***audit path <pathname>***" entry in the config.zim file. Once again, the creative use of multi-level directories addressed this problem. By creating the transaction journalling directory as a multilevel directory, a user account could be set up that would permit multilevel use of the DBMS.



Note 1: This represents a multi-level directory to support 2 labels. These labels would be an UNCLASSIFIED label (L30.1) and a CONFIDENTIAL label (L180.1)

Home directory: /usr2/testusr1

config.zim:

```
database path /usr2/testbed/pseudo-db
work path /usr2/testusr1/work
audit path /usr2/testbed/transactions
audit updates yes
•
•
•
```

areas.zim:

```
0001 /usr2/testbed/actual-db
0002 /usr2/testbed/actual-db
•
•
•
```

Figure 2: Sample Directory Structure

9.0 CONCURRENCY CONTROLS - THE MULTIUSER LOCK FILE

A review of multi-user ZIM under UNIX System V indicated that the concurrency control mechanism would be a problem under SV/MLS. This is a multi-user locking scheme based on all processes having read/write access to the *zimlock.zim* file, which is resident in the database directory. A suitable mechanism within the DBMS was needed but this was not a problem which could be addressed within the scope of the project.

The problem associated with multi-user ZIM and the *zimlock.zim* file could be addressed, in terms of a "work-around", through the use of a multilevel database directory. This directory would contain subdirectories for each level associated with the application and each subdirectory would contain a separate copy of the lock file. A ZIM configuration file, the *areas.zim* file, can be used to point to specific directories for specific tables on a table by table basis. This file would be used in this scenario to point to each actual ZIM database table file, which would be located in conventional directories.

The issue of a lack of a guaranteed rereadability was tested. The only problem that was encountered was when the following scenario occurred:

a. a set was selected by a CONFIDENTIAL process;

b. an UNCLASSIFIED process updated a table that was part of the set selected by the CONFIDENTIAL process; and

c. the CONFIDENTIAL process attempted to process the previously selected set.

The ZIM DBMS issued several error messages related to pointer and read errors to the CONFIDENTIAL process since file pointers in the temporary working file were invalid. These error messages could be trapped in the application program and the program could be reexecuted.

This approach is not a solution as it will not guarantee rereadability and will not ensure integrity across security levels since separate copies of the lock file exist for each level. The ideal solution to this problem would be the procurement of a true secure DBMS but this course of action would require new procurement actions and would cause significant project delays. It was accepted by project management that the loss of guaranteed rereadability for processes reading tables from lower sensitivity levels was an acceptable loss of functionality, given the predominantly read-only nature of the of the classified components of the applications within the project.

10.0 ZIM PROGRAMS

There are two aspects to the manner in which ZIM uses programs which assist in the building of a multilevel application. The first, the labelling of specific program files, has already been discussed. In the context of the ZIM DBMS, the inability of the DBMS to read a program will result in a warning message, which can be disable, and continued processing.

The second aspect is the possibility of building separate applications based on the sensitivity level of a given user. The ZIM data dictionary points to the location of all interpreted ZIM program files. All compiled ZIM programs are located in the *zim0001.ws* directory. It is possible to put the application programs in a multilevel directory so that a complete application is present for each level of sensitivity of the application. To the ordinary user, there will appear to be only one program directory. The filenames referenced in the ZIM data dictionary will, if they refer to multilevel directories, be interpreted by the operating system to point to the appropriate directory for the user's current security level. Document filenames, defined as absolute path references, will always point to the appropriate directory since the ZIM data dictionary will reference the appropriate multilevel directory under SV/MLS.

PART IV - CONCLUSIONS

In conclusion, this paper outlines, in terms of a specific project, how multilevel multiuser applications can be developed for an untrusted DBMS on a TCB and use the controls implicit in the TCB. The use of features implicit in UNIX SV/MLS can assist in the use of an untrusted DBMS. The specific case of the ZIM DBMS and its constraints, within the operational context of a project, demonstrate a specific means of implementing a multiuser multilevel application using untrusted DBMS on a TCB.

REFERENCES

[1] "System V/MLS 1.1.1 Trusted Facility Manual", AT&T, 13 June 1989.

BUILDING A MULTI-LEVEL SECURE TCP/IP

Deborah A. Futcher
The Wollongong Group
2010 Corporate Ridge Dr
Suite 550, McLean, VA 22102

Ron L. Sharp
AT&T Bell Laboratories
Rm 14E-214, 1 Whippany Rd
Whippany, NJ 07981

Brian K. Yasaki
The Wollongong Group
2010 Corporate Ridge Dr
Suite 550, McLean, VA 22102

ABSTRACT

This paper describes changes made to a networking protocol in order to make it "*trusted*" in a multi-level secure operating system. The protocols are the standards used by the Internet; the Transmission Control Protocol and the Internet Protocol (TCP/IP). These protocols are currently used in many heterogeneous networking environments. This paper is based on actual work being done by AT&T Bell Laboratories and The Wollongong Group in the joint design and development of a secure TCP/IP.

INTRODUCTION

The Transmission Control Protocol (TCP) and the Internet Protocol (IP) were originally developed for the ARPANET. Together they comprise one of the most popular transport and network layer protocol suites in use today, particularly within the U.S. Department of Defense (DOD). Since TCP is always run on top of IP the two are commonly referred to as TCP/IP. Initially TCP/IP provided no security services except for reliable delivery and integrity checksums. A sensitivity label was added as a possible option in the IP datagram header to enhance security. Since Multi-Level Secure (MLS) systems and networks are just now becoming available, most implementations of TCP/IP do not include this IP option.

Just adding an IP security label to each IP datagram does not provide enough security information for an MLS system. Many conditions must be met when importing information into an MLS system. Is the data labeled? Can the label be trusted to be correct? Is the host authorized to handle the level of sensitivity represented by the label? These questions and others must be answered prior to bringing networking data (i.e., IP datagrams) into an MLS host or passing it on to another network.

AT&T Bell Laboratories and Wollongong have teamed up to develop a security enhanced TCP/IP. This new TCP/IP, referred to as MLS/TCP, is fully compatible with existing TCP/IP implementations. Additional features have been added to provide network labeling and other security services in concert with System V/MLS.[1] System V/MLS is a multi-level secure enhancement to AT&T's System V UNIX® operating system. System V/MLS received a B1 rating from the National Computer Security Center in September 1989.

In addressing the problem of how to add security to a TCP/IP protocol stack, we were concerned with three non-security requirements. The first was that the specifications for the networking protocols could not be modified. This would ensure that the multi-level host would still be interoperable with all the other TCP/IP implementations. Second was that the MLS/TCP host should be able to remain trusted in an environment where both non-secure and multi-level secure hosts were part of the network. This would provide a transition path from a partially secure network (mixture of trusted and non-trusted hosts) to a completely multi-level secure network. The third requirement was that we wanted current applications to be reused without any changes (i.e., be binary compatible). This would allow "*commercial off the shelf*" (COTS) software to still be used. This requirement was later limited to those applications that did not require "*root*" privileges.[1] Since "*root*" privilege implies trust, we did not believe that having to modify a trusted application to recognize the security policy was excessive.

This paper provides some of the insights gained and lessons learned while enhancing TCP/IP to work in an MLS environment. Enhancements to the TCP/IP implementation are described. Two types of IP labels are supported and discussed in the Packet Labeling section. Changes to the route selection mechanism are also discussed. A decision was made to support trusted and untrusted application level servers and the impact to these servers is shown. The Network Interface section discusses the changes required to interface to trusted and untrusted networks. As stated earlier, some changes were required to support trusted applications. A section is included which describes some of these changes. Finally, the auditing requirements for a multi-level secure TCP/IP are reviewed.

MLS REQUIREMENTS

Introducing TCP/IP into an MLS environment places additional requirements on the implementation. Modifications are needed to provide the additional security features required to protect the data from compromise or corruption. In addition, a careful examination of the TCP/IP software must be performed to ensure that it meets the assurance requirements for an MLS system.

One of the most important requirements is the added trust that is required. Most TCP/IPs are implemented in the kernel[2] and thus have access to all of the kernel data structures. A malicious implementation of TCP/IP could violate the security policy by manipulating critical operating system data. Of course this threat is not unique to MLS hosts or even to UNIX hosts. Untrusted software in an operating system can render any security control useless; however, on an MLS host the potential damage posed by such a threat is even greater.

All data in an MLS system must be labeled. Without a label the host can not make access control decisions. There must be a strong link between the data and its associated label. The Trusted Network Interpretation[2] ("Red Book") has the following requirement concerning network labeling:

> "When the TCB exports or imports an object over a multilevel communications channel, the protocol used on that channel shall provide for the unambiguous pairing between the sensitivity labels and the associated information that is sent or received."

There is no one standard format for a sensitivity label. In addition, there are many different representations of the fields within a label. Therefore a robust implementation of an MLS TCP/IP must understand and be able to map between these multiple formats and representations.

Most implementations of TCP/IP do not handle labels. They are used on single-level networks where there is no need for labeling. Backward compatibility requirements dictate that the MLS host should be able to connect to such a single level network, accept data and associate the proper label with this data.

Networks connected to an MLS host may be accredited to handle multiple labels or only one label. The TCP/IP must ensure that no data is sent to a network that is not authorized for that data. In addition, all incoming data must be within the sensitivity range authorized for the host.

As with any protocol, TCP/IP buffers data until the receiving host can receive it or until the user is ready to read it. It is critical that the MLS TCP/IP maintain strict separation of this data inside the kernel allowing no accidental mixing of data of two different sensitivities.

All security relevant events must be audited. This includes successful and failed connections as well as any change in security parameters. Since the operating system may never see a failed connection, such auditing must be performed within TCP/IP.

1. The concept of "*root*" privilege in the UNIX environment means that the process has the capability to bypass most security checks.
2. The kernel is the part of the UNIX operating system that is separated from the user application by a distinct address space. It handles access requests to all system resources such as terminals, disks, printers, and networks.

PACKET LABELING

IP implements part of the network layer of the Open Systems Interconnection (OSI) Reference Model. IP is based on the datagram model. In this model, each data unit is treated as an isolated entity. All the information, such as a sensitivity label, necessary to transmit the data unit through the network is contained within the packet. IP datagrams contain a header which includes the source and destination addresses for the datagram and any other information that the network may require in order to transport the datagram from source to destination. Additional information can be included in the header in the form of IP options. The total amount of space that can be used by *all* the IP options sent in a datagram is limited to forty octets.

It is easy to see that a sensitivity label represented by human-readable ASCII characters could exceed forty octets in length. Thus security related information that is transmitted as an IP option is usually represented by numbers and not letters. Another reason for using numbers instead of letters is that label comparing is less costly. The computer resources required to compare two numbers is significantly less than that used when comparing two character strings.

Current IP Security Options

The Military Standard 1777 (MIL-STD 1777) specifies the Internet Protocol. Included as part of that document is a section on the defined IP options. There is a definition for an IP Security Option which includes fields for a security level, compartments, handling restrictions and transmission control code. Request For Comment 1038 (RFC 1038), currently in draft form, specifies changes to MIL-STD 1777 regarding two IP security options. The options are referred to as the "Basic Security Option" (BSO) and the "Extended Security Option" (ESO).

Basic Security Option

RFC 1038 has the following to say about the purpose of the DOD Basic Security Option.

> "This option identifies the U.S. security level to which the datagram is to be protected, and the accrediting authorities whose protection rules apply to each datagram."

The BSO defines four security levels: "Top Secret", "Secret", "Confidential" and "Unclassified." It also identifies four accrediting authorities. The BSO option reuses the option type 130 which changes the definition of the option as defined by MIL-STD 1777. MLS/TCP supports the BSO and allows the security administrator to define the meanings of the security levels.

Extended Security Option

There were concerns that the BSO did not provide all of the label information that was needed. In response to this concern a flexible security option was created that allows a recognized authority to define the contents of the option. RFC 1038 specifies the DOD Extended Security Option as follows:

> "This option permits additional security related information, beyond that present in the Basic Security Option, to be supplied in an IP datagram to meet the needs of registered authorities. If this option is required by an authority for a specific system, it must be specified explicitly in any Request for Proposal".

The ESO uses IP option type 133. See reference [3] for a detailed definition of each option. Due to the largely undefined nature of the ESO, we have chosen not to implement this option in the first release of our product.

Commercial IP Security Option

The Trusted Systems Interoperability Group (TSIG) [3] has proposed a new IP security option that better meets the

3. TSIG is composed of a group of vendors developing secure operating systems. They are working together to solve interoperability issues with respect to MLS networking.

requirements of transmitting security related information in an IP option in an open systems environment. The BSO and ESO are administered by the U.S. Department of Defense and meet defense department requirements. These requirements do not always satisfy those found in the commercial or open systems environments.

The Commercial IP Security Option (CIPSO) permits security related information to be passed between systems within a single Domain of Interpretation (DOI). A DOI is a collection of systems which agree on the meaning of particular values in the security option and which have a common security policy. The format of the CIPSO option is shown below.

8 bits	8 bits	32 bits	8 bits	8 bits	? bits		8 bits	8 bits	? bits
134	6 - 40	1 - 0xffffffff	1-255	1-34	?	• • •	1-255	1-34	?
option number	option length	DOI	tag id	tag length	info field		tag id	tag length	info field

The option length is the total length of the CIPSO option including the number and length fields. The Domain of Interpretation field is 4 octets in length. The remainder of the option is variable in length and contains a stream of tags. These tags are used to transmit additional security information associated with the datagram. TSIG has currently defined two tag types.

The first tag type is referred to as the "*bit-mapped*" tag type. Its format is shown below.

8 bits	8 bits	8 bits	0 - 248 bits		
1	3 - 34	0 - 255	bit 1	• • •	bit 248
tag type	tag length	level	bit map of categories		

The tag type is equal to 1. The tag length is the total number of octets including the tag type and length fields. The bit map can range from 0 to 31 octets in length. If bit N is a 1, then category N (as defined by the DOI) is part of the sensitivity label for the datagram. If bit N is a 0, then that category is not part of the label.

The second tag type is referred to as the "*enumerated*" tag type. It is used to describe large but sparsely populated sets of categories. Its format is shown below.

8 bits	8 bits	8 bits	8 bits	16 bits		16 bits
2	4 - 34	0 - 255	0 - 255	cat 1	• • •	cat 15
tag type	tag length	level	flags	list of categories		

The tag type is equal to 2. The tag length includes the tag type and length fields. The flags field is interpreted as follows. If the least significant bit is a 0, then all the enumerated categories are part of the sensitivity label. If the bit is a 1, then all categories defined by the DOI are set excluding the ones listed. All other bits in the flag field are reserved for future use. Each enumerated category is 2 octets in length. This allows from 0 to 15 enumerated categories per CIPSO.

With the backwards compatibility requirement, MLS/TCP allows both BSO and CIPSO security options to be used. They can be used in any combination. The security administrator for the host determines the configuration of which IP security options to use for each network interface.

Label Mapping

The method of converting a human-readable sensitivity label to machine representation is a local issue. Each host is free to use any conversion it wants. Most implementations just create a mapping table where the human-readable security attribute is converted to a number. An entry is made in the table for every legal value for each security attribute defined in the host.

The use of numbers to represent security attributes introduces a new problem when used in the environment of networked computers. It is now necessary for each host that communicates with another host to use the same security attribute to number mapping conversion. One solution is that each host has a mapping table for every host it wishes to communicate with. This introduces the problem of maintaining a large number of mapping tables when the number of hosts grows large. Another solution is to have each host connected to the network use the same global mapping table. But this solution implies that all the hosts belong to the same security domain. These solutions represent the two extreme cases.

The CIPSO option avoids this problem through the use of a flexible yet manageable solution. In most situations, when a host joins a network, it will communicate with a set of hosts with which it has the requirement to share information. Since the set of hosts will be sharing information, the security policy regarding the protection of the information should be the same. Thus for each different group of hosts sharing information, a new Domain of Interpretation (DOI) is created. If all the groups share the same security policy, only one DOI is required. The DOI in the CIPSO option is then used to point to a mapping table that is common to all the hosts using the same DOI or within the same security domain of interpretation. MLS/TCP can support multiple DOIs for hosts that belong to more than one security domain such as gateways.

ROUTING

When a host is connected to a network, the security policy may state that data labeled at a certain security level is restricted to a particular path it takes through the network. IP normally chooses the least cost path, where cost is the number of hops that an IP datagram would traverse. TCP uses the datagram service provided by IP. TCP provides for the reliable delivery of a stream of data from source to destination. By using the services of IP, TCP will gain some of the datagram capabilities. One such capability is that IP will chose the path that an IP datagram takes dependent upon the current conditions in the underlying network. Thus if one gateway along a path goes down, IP could detect the problem and choose to route IP datagrams through a different path. Figure 1 depicts this situation. If host A wishes to communicate with Host B Secret information then it must use Net 1 or Net 3. If the routing policy does not take labels into account then the connection could be set up through Net 2. TCP will have provided the service requested but the security policy will be violated.

Thus, the algorithm that IP uses to determine the path that the datagram takes required modifications to make it cognizant of sensitivity labels. This change required that each physical network interface connected to the host be assigned a range of sensitivity labels. IP compares the label of the packet to be sent to the network label range. If the packet label is not within this range then that path will not be chosen.

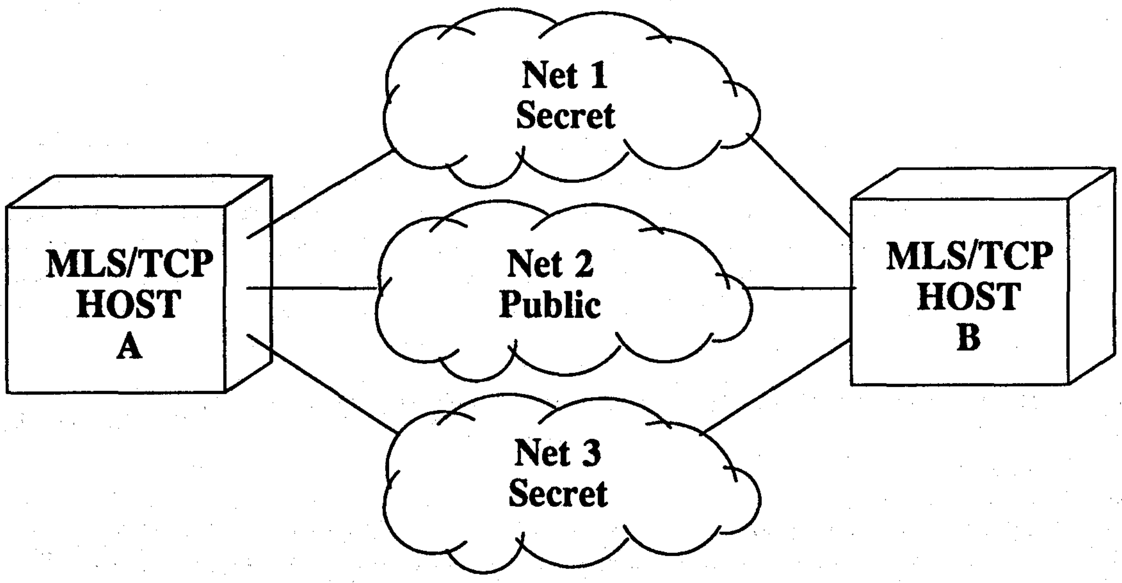


Figure 1: Multi-Level Secure Routing

NETWORK SERVERS

Network applications are sometimes described by a client/server model. The client and server together implement a defined application layer protocol. The client is the application that is requesting some service while the server is the application providing that service. The three most widely known network applications are the File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and the TELNET Protocol (TELNET).

Servers normally accept connections from any host. Each service has assigned to it a unique "*well known*" port number. Using this port number, the server will notify TCP that it is willing to accept any connection requests to its port number. This is commonly called a "*passive open*." When a server posts a passive open, the TCP state for that connection is in the "LISTEN" state. Thus servers may also be called "*listeners*." The client application knows what service it is requesting on behalf of a user. With this information, it can look up the corresponding well known port number for that service. The client then makes an "*active open*" to the server's well known port number. The server gets notification from TCP that a client is requesting a network connection. The server can accept or reject the request. If accepted, the network connection is established and the client and server can then communicate. Server processes will normally spawn[4] a child process and it is the child process that will perform the work requested by the client. The parent process is then free to go back to listening for new connection requests.

Untrusted Servers

When any server process requests that a passive open be performed, the networking software checks to see if the process has "*root*" privileges.[5] If it does not, the networking software stores the sensitivity label of the server process as the "*session label*." This action is taken without any assistance from the server process. The session label is used to restrict all incoming connections to the server to have the same sensitivity label as the untrusted server process.

When a client connection request comes in, the networking software checks to see if the session label is set. If so, then a label compare of the sensitivity label from the IP datagram of the incoming connection request is made against the session label. If the labels are equivalent, then the rest of the processing for connection establishment is performed. If not equivalent, the client's connection request is rejected. This allows untrusted servers to be supported without modification while restricting their operation to a single label.

There is a generic problem with untrusted servers; any user of the system has the capability to create an untrusted server. This allows the import/export of data, albeit at a single level, without any identification or authentication processing being performed. We solved this problem by restricting all server executables to be stored at the "*system low*" level, a level at which normal users can not create executables.

Trusted Servers

When a process with "*root*" privileges requests a passive open, the networking software does not fill in the session label. When a client connection request comes in, the networking software detects that there is no session label set for the associated listener. The networking software then checks the sensitivity label of the incoming IP datagram to make sure that it is within a range of values that the security administrator has set for the host. If within range, the networking software notifies the server of the connection request. If the server accepts the connection request, the session label for the new connection is set to the label of the incoming IP datagram. This allows the trusted server to know the sensitivity label of the client process.

The server process spawns a child process. This child process still has "*root*" privileges. Before the child execs a program that will provide the service, it must perform a few tasks. First it must retrieve the session label from

4. In the UNIX environment, a new process is "spawned" by executing the system call "fork." This creates a new process that is an exact copy of the original process. It has the same security privileges and has access to all the same open files. The new process can then use the system call "exec" which overlays the current running process with another program if it wants to run a different program.

5. In a MLS UNIX environment an "*untrusted server*," is any server process that does not have "*root*" privileges.

TCP and change the process sensitivity label to the session label. The child process must then change its User Identification (uid)[6] from "*root*" to another uid thus removing its "*trusted*" capability. Only after the child process has removed its "*trusted*" capability can the child process exec the program that will provide the requested service. In this way the program that provides the actual service does not need to be trusted. The uid that is selected can be predetermined based on the service the server is providing. For example if the server is providing the Simple Mail Transfer Protocol service, the uid is that of the "*mail*" daemon. Other services require that some other authentication mechanism be performed. For example the TELNET server relies on the supplied /bin/login[7] program after setting up a terminal environment.

There maybe cases when a server needs to be trusted in order to gain access to other system resources but it only wants to accept network connections at a specific session label. A trusted server is allowed to make a call to TCP that will set the session label. Thus when a client connection request is received, the networking software detects that the session label is set and processes the request as if the server were untrusted.

MODIFIED NETWORK APPLICATIONS

It was previously mentioned that the three most widely known network applications are implementations of the TELNET, FTP and SMTP protocols. This section goes into more detail on how the implementation of each network application had to be modified to be supported under MLS/TCP.

TELNET

The TELNET client application required no changes. This is due to the fact that the kernel TCP software can obtain security relevant information about the user of a client TELNET without any assistance from the TELNET program. The TELNET server also required no changes. The reason for this is that server TELNET is implemented in the kernel and it ultimately depends upon the /bin/login trusted program to perform the UNIX login processing.

FTP

The FTP client application required no modification. In order to support the server FTP program, two changes were required. First a trusted front-end to server FTP was created. This trusted program performed the identification and authentication portion of the FTP protocol. The FTP protocol for identification and authentication requires a user name and the password associated with the user name. After checking that the user name and password are valid, the trusted front-end makes a call to TCP to retrieve the session label. A check is then made to see if the user name has been authorized to process information at that session label. If not, an error is returned to the client FTP and the network connection is closed. If allowed, the trusted front-end changes the security attributes of the process to match those of the session label. It then execs the "*original*" FTP server program. The original "*untrusted*" FTP server program has been modified to disable the user name and password commands. The original FTP server remains an untrusted program that responds to the commands requested by the FTP client. The untrusted server FTP process can only access correctly labeled data because of the MAC and DAC checks performed by System V/MLS.

SMTP

The SMTP protocol application presented a different set of security considerations due to the fact that it is most often implemented and accessed on behalf of the user via the general internet mail routing application known as "*sendmail*". As a stand-alone protocol specification, SMTP as its name implies provides for a very simple set of handshaking and etiquette requirements. In contrast, the sendmail application is a complicated program which integrates the SMTP protocol implementation with such functions as mail collection, routing and queuing.

6. UNIX assigns a user identification number to each user account. The "root" account has always used the uid of 0. Thus a non-zero uid implies some user that does not have the trust associated with "root."

7. /bin/login is a trusted program used by UNIX to perform identification and authentication.

On the client side, no real modifications were necessary for the main processing path. A user wishing to send mail to a remote system uses the MLS mail interface program which in turn invokes sendmail to route the message to the remote destination. If sendmail determines that the SMTP protocol should be used to accomplish this task, it attempts to establish the connection. The networking software will automatically set the session label to the user's current operating level. Assuming the connection can be established and their are no violations of the "*simple*" protocol requirements, the message is delivered to the remote system and stored in a user mailbox file whose label matches that of the sending user. The problem arises when something goes wrong in this scenario such as the inability to connect to the remote system. In this case, sendmail queues the message for later delivery attempts. Queue processing in an MLS environment adds an additional complication to the sendmail application. The solution was incorporated into the trusted server section of the program.

Most of the changes made to the server side of this application in support of the MLS/TCP environment are similar to those already described for the other trusted server applications. Specificly, when invoked as a server, sendmail first verifies that it is executing with "*root*" privileges. If so, it sets up a trusted SMTP listener without an associated session label set. Subsequent attempts by client SMTP applications to establish connections are handled by spawning new processes which set the label of the server to match that of the incoming connection and the uid of the process to the uid of the mail daemon before continuing with the normal SMTP transfer function. This differs somewhat from the "identification and authentication" process described for the FTP and TELNET protocol applications as the uid is automatically set to the uid of the mail daemon. However, since mail accepted by sendmail's SMTP server will be delivered to the local user at the level at which the sending user invoked SMTP, if the local user is not authorized to operate at that level, he will not have access to the message. In fact, the MLS mail interface program will not even notify him that it exists.

As mentioned above, a second change to the sendmail server software was necessary to handle the queue processing. Standard sendmail implementations spawn a new process which retains "*root*" privileges to periodically examine the mail queue and attempt to deliver any accumulated messages. The server was modified to create a separate process for each level at which mail capability has been authorized and set the label of each process to match. The user id of all of these processes is set to the mail user ID. This solution was chosen because it reduced the amount of processing that required root privileges while minimizing the changes to the existing sendmail implementation.

NETWORK INTERFACE

One of the strengths of TCP/IP is that it can connect to many different types of networks. Some of the common types are 802.3 (Ethernet), token ring, X.25, or even a serial RS232 line. A secure TCP/IP can protect the data only while it is in the host. Once it leaves the host it is the responsibility of the network to protect it. For many Local Area Networks (LANs) this protection is just physical control of the communications media (the copper wire). For other networks there are devices that provide special security services such as encryption or mandatory access control. Each of these types of network interfaces have unique requirements pertaining to security. A secure TCP/IP should be configurable to handle the needs or shortcomings of these networks.

Trusted Networks

Within the context of this paper, a trusted network is one in which the security parameters provided are guaranteed to be accurate. These parameters may be provided by a network device or by the host at the other end of the connection. It must not be possible for a non-trusted host or user to be able to interject false security parameters into a trusted network.

One example of a trusted network is the Verdix VSLAN network.[8] The security parameters (i.e., the sensitivity label) for each 802.3 packet is provided by a network interface card. We have modified an 802.3 network driver to accept the VSLAN label and convert it to a CIPSO or BSO label. The label and the data packet is then passed up to IP. IP attaches the label to the packet and sends it up to TCP or out another network interface depending

8. The VSLAN network was evaluated by the National Computer Security Center and has received a B2 rating.

on the IP destination address.

Another example of a trusted network is Blacker. The Blacker Front End (BFE) is a device that provides data confidentiality through the use of high-grade encryption. Blacker uses BSO to obtain the security level of data and performs access control based on this label. No additional changes were required to support Blacker.

A simple Ethernet network can be a trusted network if all hosts on the network are trusted. The level of trust provided by this network is equal to the level of trust of the least secure host on the network. For this network, the security parameters are passed in the TCP/IP protocol and a separate security interface is not required.

<u>Untrusted Networks</u>

Most networks in use today offer no security services and any security parameters provided by these networks can not be trusted. We could require MLS hosts to only connect to MLS networks, however that would not be practical. Our solution was to assign a fixed set of security attributes to these networks. These attributes are provided by the security administrator of the MLS host and reflect the security attributes associated with the untrusted network.

As mentioned earlier, all data coming into an MLS host must have a label. Datagrams from untrusted networks should not contain a label. If a label is present in the datagram then it can not be trusted. Our solution is to insert a CIPSO or BSO label into the IP datagram as it enters the MLS host. If a sensitivity label is already present in the datagram it is overwritten with the new label. This sensitivity label is obtained from the fixed set of security attributes assigned to that network. Figure 2 illustrates the function of a MLS/TCP gateway between an untrusted, non-labeled network and a trusted, MLS labeled network. If the label was not added then hosts on the untrusted network could not communicate with hosts on the MLS network where a label is required.

For packets going from the MLS host to the untrusted network we provide the capability to "*strip*" out the CIPSO or BSO label from the datagram. Some of the hosts on the untrusted network may not be able to handle an unrecognized IP option and may crash the host.

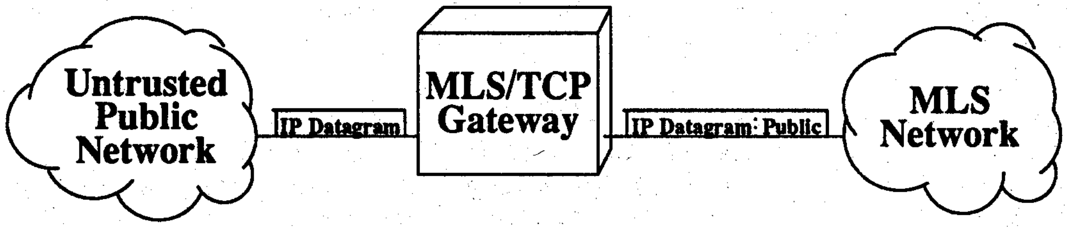


Figure 2: Label Insertion and Stripping

<u>AUDITING</u>

All security relevant events must be audited. A security relevant event is any action taken by a host or network which provides a user with access to a resource or that effects a change to security information. The information included in an audit record must be sufficient to determine the characteristics of the access or change. Below is a list of some of the events that are recorded.

1. All failed or successful connections to the host
2. Incoming packet with a label outside of the host label range
3. Outgoing connection refused due to no route found that meets security requirements for the level of the requested connection
4. Label on incoming packet contains a security level not recognized
5. CIPSO DOI on incoming connection not supported by the host
6. TCP connection closed

These new audit records are included in the host audit record. Some TCP/IP events could generate a large amount of audit records and overwhelm the system. For example, all audit events at the packet level could generate an audit record for every packet associated with a particular connection. For this reason we have included the capability to allow the security administrator to turn off the recording of any event that the administrator determines is not needed.

An argument could be made that all packets received should be audited. As mentioned above this would quickly consume all the disk space on the system. Since TCP is a connection oriented protocol, we feel that just auditing the success or failure of the connection is enough. The operation of connecting to a remote host using TCP is analogous to opening a file. The Orange Book[4] requires the file open to be audited, but does not require auditing of the individual reads or writes. Likewise auditing the closing of a file is also required and TCP audits the closing of each connection. UDP (User Datagram Protocol) is a connectionless protocol and audit of each packet would probably be required. The networking software does not currently implement a trusted UDP but one is planned for a later release.

ASSOCIATION WITH THE TNI

The Trusted Network Interpretation (TNI), also known as the "Red Book" describes the requirements for MLS networks. It recognizes that most networks are made up of many components each of which may provide a different security service. For this reason the TNI breaks the requirements for a secure network into four distinct areas. These areas are Identification and Authentication (I&A), Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Audit. The implementation described in this paper is designed and implemented to satisfy the MAC and Audit requirements. It is expected that the DAC and I&A requirements are satisfied at a higher layer in the protocol stack.

CONCLUSIONS

Despite the complicated nature of the MLS requirements, the design and implementation of this project went very smoothly. TCP/IP already embodied many important concepts such as data separation and integrity. The flexibility of the options in the IP header was a particularly critical ingredient. Many of the changes involved hooks in the TCP/IP that called new operating system routines. There were no major rewrites of TCP/IP or UNIX code.

Most of the new capabilities have been embedded in the internal workings of TCP/IP and can not be seen outside of the host or even by the user. The only change seen outside of the host is the newly supported IP security labels and those can be stripped if not needed. The user application interface to TCP/IP has not been changed so all applications should continue to operate. Some additional applications interface features were added to support trusted applications that understood labeling.

REFERENCES

1. C.W. Flink and J.D. Weiss, "System V/MLS Labeling and Mandatory Policy Alternatives", Proceedings of the 1989 Winter USENIX Conference, February, 1989.

2. National Computer Security Center, "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria", NCSC-TG-005, 31 July 1987

3. M. St. Johns, "Draft Revised IP Security Option", Request For Comments 1038, January 1988.

4. Department of Defense Standard 5200.28-STD, "Department of Defense Trusted Computer System Evaluation Criteria", December 1985

THE CASCADE PROBLEM: GRAPH THEORY CAN HELP

John A. Fitch, III[1] and Lance J. Hoffman

Department of Electrical Engineering and Computer Science

George Washington University

Washington D.C. 20052

Abstract

This paper presents a new approach, based on finding shortest paths in a graph, for solving the cascade problem. The result is an efficient ($O(N^3)$) algorithm, where N is the number of security domains in the network. The paper provides background on the cascade problem, generalizes the problem from its traditional military roots, and then applies the shortest path technique to a military example. The shortest path approach appears quite general and provides a method based on established mathematics for evaluating network security.

Keywords: Cascade problem, graph theory, shortest path, network security, risk analysis.

1. The Cascade Problem

The cascade problem was first defined and discussed in [14]. The importance of the cascade problem is that it demonstrates how networking systems together may produce unacceptable risks even though the individual systems in the network are secure and reasonable interconnection rules are followed. "Reasonable interconnection rules" means that the network connections comply with security policy and are secure from external attacks such as wiretapping. This paper provides background information on the cascade problem, generalizes the problem from its traditional military roots, and applies a resource-constrained shortest path technique to a military example. The result is a new, efficient ($O(N^3)$) algorithm, where N is the number of security domains in the network, for determining if a network has a cascade problem. This graph-theoretic approach appears quite general and provides a method based on established mathematics for evaluating network security.

1.1. The General Cascade Problem

The cascade problem is described in [14, 8]. Both references focus on cascading in military networks where both security risk assessment and system security evaluation use Defense Department standards and guidelines. This section describes the cascade problem in more general terms, provides the background information to understand the types of networks in which cascading may be a concern, and presents a military example of the cascade problem.

1.1.1. General Cascade Problem Definition

The cascade problem belongs to a subspace of the problem set that asks, "If secure systems are connected together, is the resulting network secure?". This section partitions the problem set to place the cascade problem in perspective and then presents a more formal definition of cascading.

Before partitioning, it is first necessary to define *secure system*. This paper defines a secure system as a system that has undergone both a system security evaluation and a risk analysis evaluation that results in an acceptable risk of operating the system. A risk analysis considers the assets of a system and threats against it to determine how much security is sufficient. System security can be modeled as a

[1]Mr. Fitch is also affiliated with GTE Government Systems

function of several parameters: physical security, personnel security, administrative security, communications security, and computer security [15]. These parameters can be represented by classes of countermeasures that reduce system risks. For example, physical security can be described by the class of countermeasures that includes locks, fences, and guards. A system security evaluation, therefore, measures the effectiveness of the countermeasures used in the system.

The first partitioning of the network security problem space is to divide the space into networks that (for security purposes) can be treated as a single system and networks that cannot be treated as a single system. Cascading is only a concern in the latter type of network. There are reasons why some networks cannot be or are not viewed as single systems. First, the network may be so large that a single system security evaluation is not feasible, so a divide-and-conquer approach must be taken. Second, the network may be made up of systems that are owned or operated by differing administrative entities or systems that use different system security evaluation or risk assessment methods.

Having limited the problem space to networks that either are not or cannot be evaluated as single systems, the next step is to reduce the problem space by examining the conditions under which two systems would interconnect. As a minimum, the administrators of the two systems have to mutually agree that the other system is secure in its own environment; that is, they need to understand and accept the risk assessment and security evaluation methods used by the other and believe that the analysis was done correctly. This does not imply that all the systems on the network are equally secure: it means that each system recognizes that the other's security is good enough as a stand-alone system (that is, before the interconnection is considered). If one system does not believe that the other is secure, then there is a clear risk to sharing data with that system. For example, two systems that implement completely different security policies or conduct very different evaluation methods are unlikely to share sensitive data. For the cascade problem, only mutually recognized secure systems are interconnected and each system provides the other with its system security evaluation metrics.

Having mutually recognized that the other system is independently secure, the next step is to decide which assets (or classes of assets) are to be shared with the other system. This step is closely related to mutually accepting the other system's security evaluation because each system must identify a subset of assets for export that it believes the other system will protect accordingly. This does not imply that the two systems must have identical export sets: the exchange may be one way, with one system acting only as an exporter and the other acting only as an importer.

Because each exporting system believes that the importer will properly protect the exported asset, it implicitly believes that the importer will share the asset with third-party systems only if those systems are also secure. This means that a system needs to consider only the security of the system directly involved in the interconnection and not the security of all the systems in the network in order to be assured that the exported asset is properly protected. This "nearest neighbor" approach thus creates an implied transitive property of protection.

Because the systems agree to share assets via a network connection, the security of the connection itself must be addressed. The cascade problem assumes that the interconnection mechanism itself is secure; (that is, assets are not threatened when on the connection) and that the threats are only at the two systems involved in the connection.

In summary, the following type of network is being considered:

- The network consists of independent secure systems; that is, each system in the network, based on its own risk analysis and system security evaluation, is secure before considering network connections.
- For size or political reasons the network cannot be treated as a single system and undergo a security evaluation similar to that of the component systems in the network.

- Before agreeing to an interconnection, each system mutually recognizes the security of the other.
- The systems involved in a connection only share assets that the exporting system believes the importing system will protect properly.
- The connection itself is secure; that is, there is no threat posed against data while in transit between the systems.

Limiting the discussion to these types of networks, it is now possible to define when a cascade problem exists:

A cascade problem exists *when independent, mutually recognized secure systems are interconnected by secure channels to create a network system that is not secure.*

1.1.2. Why Cascade Problems Occur

The existence of the cascade problem results from several factors. The decision to allow an interconnection between systems was based only on assuring the protection of the assets being shared; it was not based on all the assets in the source and destination systems. This at first appears adequate because the two systems are independently secure, but the fact of the interconnection means that the two systems are no longer truly independent. The cascade problem exploits these two facts in a subtle fashion based on risk analysis principles.

One purpose of a risk analysis is to determine how much security is needed to protect an asset. Because the asset has some determined value, there is a threshold on the amount one is willing to spend on protection. For example, one may not be willing to spend $75 on a safe to protect a $100 watch, but may be willing to spend $20 to buy better locks for the door: there is a limit at which one accepts the residual risk to an asset rather than pay more for security. Another way to view this concept is that security is measured by the amount of effort required to steal the watch. The watch owner wants the thief to have to spend the effort to defeat a $20 lock in order to steal the watch. In a computer system, there is an analogous threshold where one is willing to accept the residual risk to the asset (such as compromise or destruction of data) rather than incur the cost of additional protection (see [16]). The definition of a secure system in the previous section is consistent with this cost/reward observation.

A penetration (either by a human or by "nature") of one of the systems may cause other systems' assets to propagate to an interconnected system. While a stand-alone secure system that suffers a penetration is, by definition, willing to accept the local penetration as within acceptable risk, that system does not necessarily accept the export of other assets as within acceptable risk. (This was the point of identifying import and export sets.) Thus the cascade problem is essentially a risk assessment problem that measures network risk based on local risk metrics of an export of data not in the export set. The problem is called cascading because the links between the systems act as conduits that cascade assets along a path between systems. If the assets arrive at a system that does not adequately protect them, then a cascade problem exists.

Thus, determining if a network has a cascade problem requires identifying if the network is of the type identified in the previous section, stating the acceptable level of risk against loss by cascading, calculating the actual cascade risk based on the network configuration, and assessing if the cascade risk exceeds the acceptable level. As in the example of the thief and the watch, security from cascading can be measured by the amount of effort required to defeat the protection mechanisms. Security from cascading can be measured by requiring a penetrator to expend a stated quantity of resources to affect the penetration(s) necessary to cause a loss via cascading. From a penetrator's perspective, cascading can be viewed as an accumulation of costs as the penetrator creates a path of penetrations through the network.

1.2. Military Cascade Example

To derive a specific cascade problem from the general cascade problem requires indicating the risk assessment and system security evaluation methods used by the systems in the network. This section briefly reviews the risk analysis and evaluation methods used in the military cascade problem as defined in [14, 8] and presents an example of a network with a military cascade problem.

The risk assessment method used in [14, 8] is based on the environment guidelines given in [12, 13] where assets values are measured by the security classifications of the data in the system and the threats are measured by the minimum user clearance in the system. The risk analysis method uses the maximum data classification and minimum user clearance as indices into a table to determine a recommended amount of computer security for the system. The amount of computer security is measured by a specific rating defined in the Orange Book [10, 11]. Figure 1-1 shows a table from [13] that maps a (minimum user clearance, maximum data sensitivity) pair to a required Orange Book level of computer security. The Orange Book computer security ratings are ordered as D < C1 < C2 < B1 < B2 < B3 < A1.

		Maximum Data Sensitivity						
		U	N	C	S	TS	1C	MC
Minimum Clearance or Author-ization of System Users	U	C1	B1	B2	B3	*	*	*
	N	C1	C2	B2	B2	A1	*	*
	C	C1	C2	C2	B1	B3	A1	*
	S	C1	C2	C2	C2	B2	B3	A1
	TS(BI)	C1	C2	C2	C2	C2	B2	B3
	TS(SBI)	C1	C2	C2	C2	C2	B1	B2
	1C	C1	C2	C2	C2	C2	C2	B1
	MC	C1	C2	C2	C2	C2	C2	C2

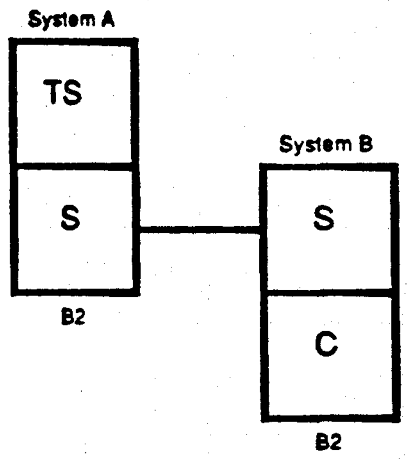
Figure 1-1: Security Index Matrix For Open Environments (adapted from [13])

The Orange Book rating is used as the computer security portion of a system security evaluation that also includes other factors, such as physical and procedural security. The cascade problem in [14, 8] considers only the computer security portion of a system security evaluation. To simplify the mutual recognition of each system's security and to follow the example from [14, 8], only the computer security portion of the system security evaluation is considered here as well.

The next step is to define the acceptable import and export sets between systems. This is done by requiring that the interconnection between systems obeys the military multilevel security policy of "no read up" and "no write down" between data at different classification levels. The classifications in the example are ordered as CONFIDENTIAL < SECRET < TOP SECRET. See [2] for details on the multilevel security policy and [3] for a general lattice-based model of secure information flow.

Having reviewed the military risk assessment and security evaluation methods, the military cascade problem can now be discussed. The cascade problem for the military multilevel system is informally defined in [14] as when a penetrator can take advantage of the network connections to compromise data over a range of sensitivity levels that is greater than the accreditation range of any of the systems that must be defeated to do so. (An accreditation range is the set of security levels a system is trusted to process and separate correctly according to the information flow policy).

The example shown in Figure 1-2 from [14] demonstrates the military multilevel cascade problem. System A has an accreditation range of (SECRET, TOP SECRET) and the minimum user clearance is SECRET. System B has an accreditation range of (CONFIDENTIAL, SECRET) and the minimum user clearance is CONFIDENTIAL. Based on the guidelines in [13] and shown in Figure 1-1, System A requires at least B2 computer security and System B requires at least B1 (System B's rating of B2 in Figure 1-2 satisfies this constraint).



TS = TOP SECRET, S = SECRET, C = CONFIDENTIAL

Figure 1-2: A Network With A Cascade Problem

Each of the systems agrees to export only SECRET information to the other. This interconnection conforms to the military information flow policy and thus defines the allowed export set.

The cascading in this network occurs by assuming a penetration of the operating system protection mechanisms at both end systems. If a penetrator compromises System A, TOP SECRET information may be leaked via the SECRET connection to system B. If system B is compromised, then this TOP SECRET information may be leaked to a user who is only cleared CONFIDENTIAL. Thus the network has a cascade problem because the penetrator has compromised three levels of data by defeating two systems with accreditation ranges consisting of two levels of data.

To determine if a network has a cascade problem, the next section formulates the multilevel military cascade problem as a resource-constrained shortest path problem.

2. Shortest Path Formulation of the Military Cascade Problem

Formulating the cascade problem as a resource-constrained shortest path problem provides an efficient algorithm for determining if a network has a cascading problem and thus improves greatly on the heuristic presented in Appendix C of [14]. The resource-constrained shortest path algorithm is also superior to the algorithm designed by Millen [9] based on matrix multiplication. There are several motivations for performing a cascade analysis. For example, a system administrator may make a decision to join or not to join a network based on the risk posed by cascading. In a network where there is additional cooperation between the system administrators, the network can possibly be re-architected to eliminate the cascade so that all parties may securely use the net.

The resource-constrained shortest path approach to determine whether or not a network has a cascade problem is based on three phases: Preprocessing, Shortest Path Calculation, and Postprocessing. The details of each of these steps is provided in the following sections.

2.1. Preprocessing Step

The preprocessing step consists of three actions:

- Defining the cascade problem as a graph by identifying nodes, edges, and weights;
- Viewing the problem from the penetrator's perspective by allocating the penetrator a set of resources; and
- Defining the resource consumption function that determines how the network consumes the penetrator's resources.

The formulation of the cascade problem as a graph begins with the definition of *protection domains.* Appendix C of the Trusted Network Interpretation [14] defines a protection domain as a (system, level) pair. The protection domains in Figure 1-2 are (A, TOP SECRET), (A, SECRET), (B, SECRET), and (B, CONFIDENTIAL). The protection domains are the nodes of the graph in the shortest path formulation of the cascade problem.

The edges in the graph are the flows between protection domains. Viewing the problem from the penetrator's perspective, edges are assigned as follows:

1. An edge between nodes (protection domains) is created if it represents a network interconnection. This edge is weighted 0 because it is an allowed flow under the military flow policy and, therefore, represents no cost to the penetrator.
2. An edge between nodes internal to the same host system is created if it represents an allowed information flow. This edge is weighted 0 because it conforms to the military flow policy and therefore represents no cost to the penetrator.
3. An edge between nodes internal to the same host system is created if the flow represents a downgrade; that is, if the flow is not allowed by the military flow policy. This edge is weighted by the Orange Book computer security rating of the host system because it represents having to defeat the computer protection mechanisms in order to achieve the information flow.
4. For mathematical completeness, flows from a node to itself cost 0 and all other node pairs receive an edge weight of infinity.

The path a penetrator can follow through the network thus consists of steps consisting of penetrations internal to a host system or a legitimate network link.

Whether or not to consider allowed flows internal to a system depends on whether the objective is to locate the core paths that actually cause the cascades (achieved by not considering flow 2 above) or to locate all information flows that may be threatened by the cascade via legitimate flows into the core cascading paths (achieved by including the type 2 flows). This paper will not apply the type 2 flows to the example problem and will thus search for core cascading paths.

Having defined the nodes, edges, and weights, the next step is to allocate a set of resources to the penetrator and define a resource consumption function. The cascade problem in [14] treats the source and destination protection domains as requiring the same level of protection as a stand-alone system;

that is, any network path of protection domains must meet the computer security protection given in the Environments Guidelines [12, 13] (see Figure 1-1). For a specific source and destination, one could use the matrix lookup to determine the required path protection and allocate that quantity of resource to the penetrator. Because the concern here is to find all cascading paths, it eases analysis to calculate the cost of all paths and then test the path cost against the required path protection as part of the postprocessing stage rather than preallocate a fixed resource to be used for path pruning during the shortest path calculation.

The consumption function used here is similar to what Millen calls the *path resistance* [9]: the cost of a path is the cost to the penetrator of achieving the information flow from source to destination protection domain. According to [14] and [8], the example of Figure 1-2 has a cost of B2 for the cascading path between the (A, TOP SECRET) and (B, CONFIDENTIAL) protection domains. This cost of a path is found by taking the maximum of the costs of the edges in the cascading path. This results in a consumption function that states that **for the military cascade problem, the amount of penetrator resources consumed on a path between protection domains is equal to the largest edge cost in the path**. Naturally, the penetrator wants to minimize the path cost between source and destination domains because it represents the level of effort required to effect a cascade. This objective (minimizing the consumption function) has now mapped the cascade problem to a resource-constrained shortest path problem.

The consumption function for the military cascade problem implicitly assumes that if a penetrator can defeat a system with a specific security rating (B2 in the example), the penetrator can defeat other systems with the same rating with no significant additional effort. By viewing the problem from a shortest path perspective, other consumption functions can be easily defined and tested. For example, if one assumes that all system penetrations are independent, then the corresponding consumption function simply sums the cost of all the edges along the path. A corporate cascade example that uses summation as the consumption function is in [4].

The network shown in Figure 2-1 is used to demonstrate the shortest path method for finding cascading paths. The results of the preprocessing step are shown in Figure 2-2. The security levels in the circles are the graph nodes and the dashed boxes indicate the domain in which the nodes belong. Note that the example network includes both one-way and bidirectional flows. The adjacency matrix for the preprocessed system is also shown in Figure 2-2.

2.2. Shortest Path Calculation

Having defined the nodes, weights, edges, and the consumption function, it is now possible to apply the shortest path algorithm. Because the objective is to first determine if the network has a cascading problem, an all-pairs algorithm is used to calculate the shortest path costs between all pairs of security domains. Should the all-pairs algorithm indicate a cascade problem, a specific source-destination shortest path algorithm can be used to yield the actual path involved. (The act of determining all the edges in the shortest path can actually be incorporated into the all-pairs algorithm, but the two steps are kept separate here for clarity.) The all-pairs algorithm presented here is similar to that of finding the transitive closure of a graph. In fact, as long as the relationship between the edge weights and the consumption function forms a closed semiring, an N^3 algorithm can be used [1] to find all paths. This is indeed the case because the computer security evaluations can be ordered as D < C1 < C2 < B1 < B2 < B3 < A1 and, therefore, mapped to integer values; and because the operations **minimum** and **maximum** needed to optimize and express the consumption function can be shown to be valid + and **o** operations, respectively, on the closed semiring of integers [5].

As a consequence, the all-pairs algorithm shown in Figure 2-3, which is modified from [6], is used. The graph is stored as an N-by-N adjacency matrix named *cost*, where N is the number of protection domains. The array *a* is the resulting N-by-N matrix of least path costs under the military consumption function defined in the previous section. The line

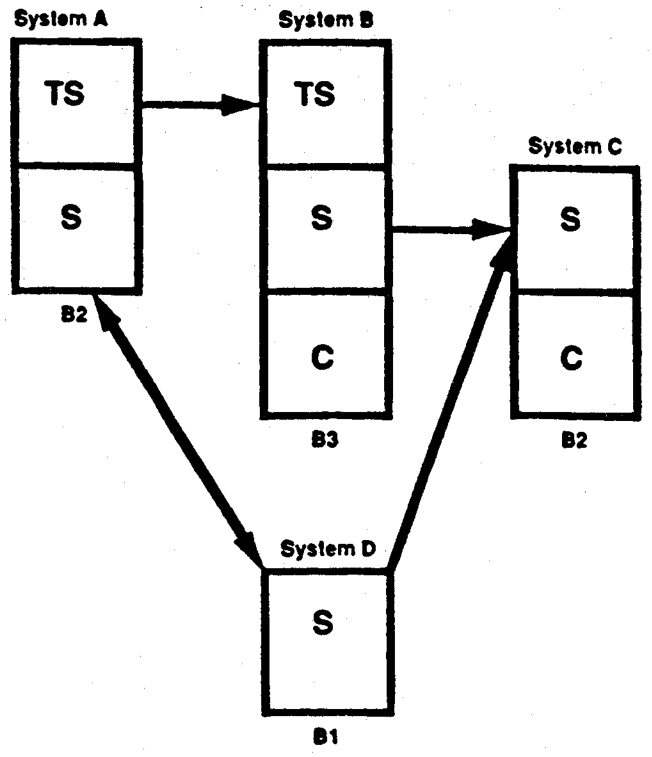


Figure 2-1: Military Network With A Potential Cascade Problem

```
a[i,j] = min(a[i,j], max(a[i,k], a[k,j]))        (1)
```

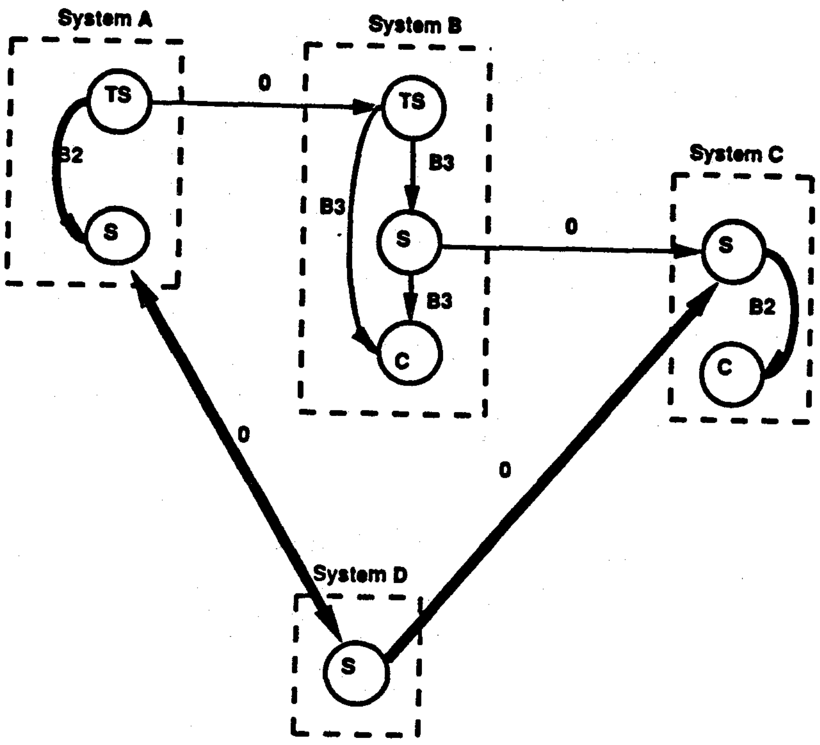
in the algorithm represents minimizing the military consumption function.

The all-pairs algorithm provides a shortest path solution in $O(N^3)$ time, although the postprocessing phase has not been considered yet. The shortest path cost results table is shown in Figure 2-4. As will be shown, the postprocessing is $O(N^2)$ so the computation complexity to determine if the network has a cascade problem remains as $O(N^3)$ where N is the number of protection domains. For comparison, Millen's work [9] is not quite as efficient. He calculates the resistance of all paths in the network by a matrix computation requiring $O(N^3 \log_2(N))$ steps.

2.3. Postprocessing Step

Figure 2-4 shows the cost of the shortest path between all pairs of (source, destination) security domains under the consumption function defined in equation (1). This path cost represents the minimal effort required by a penetrator to effect an information flow from the source to destination domain. The Postprocessing step determines whether the cost of the paths is within acceptable risk. This is done by considering the minimum user security clearance at the destination domain for all pairs of security domains. The real system risk is not that a penetrator has simply achieved a flow from one source security domain to another at an unacceptable cost; the risk is that a user who is not cleared for the information (as it was protected at the source domain) may actually obtain this information at the destination domain.

The risk acceptance test can be done as a set of table look ups for each security domain pair as follows:



Src/Dest	(Sys A,TS)	(Sys A,S)	(Sys B,TS)	(Sys B,S)	(Sys B,C)	(Sys C,S)	(Sys C,C)	(Sys D,S)
(Sys A,TS)	0	B2	0					
(Sys A,S)		0						0
(Sys B,TS)			0	B3	B3			
(Sys B,S)				0	B3	0		
(Sys B, C)					0			
(Sys C,S)						0	B2	
(Sys C,C)							0	
(Sys D,S)		0				0		0

TS = TOP SECRET, S = SECRET, C = CONFIDENTIAL

Figure 2-2: Military Network After Shortest Path Preprocessing

```
procedure allpairs(
     cost : adjacencymatrix,  {initial edge cost matrix}
     var a:adjacencymatrix,   {all pairs shortest path costs}
     n :integer)              {number of security domains}
{ Computes the shortest path cost between all pairs of domains}
{ cost[1..n,1..n] is the initial graph cost adjacency matrix }
{ a[1..n,1..n] is the cost of shortest path between nodes }
var i : integer;  {loop control for source nodes}
    j : integer;  {loop control for destination nodes}
    k : integer;  {loop control for intermediate nodes}

begin
  {copy the array cost into the array a}
  for i := 1 to n do
    for j = 1 to n do begin
      a[i,j] = cost[i,j];
    end;

  {Calculate shortest path cost for all domain pairs}
  for k = 1 to n do           {for path with highest node index k}
    for i = 1 to n do         { for all possible source nodes}
      for j = 1 to n do       {for all possible destinations}
        {if i->k->j cost is smaller than current i->j cost}
        {then update the i->j path cost.  In other words, }
        {minimize the consumption function}
        a[i,j] = min(a[i,j], max(a[i,k], a[k,j]))
 end.
```

Figure 2-3: All-Pairs Shortest Path Algorithm (adapted from [6])

Src/Dest	(Sys A,TS)	(Sys A,S)	(Sys B,TS)	(Sys B,S)	(Sys B,C)	(Sys C,S)	(Sys C,C)	(Sys D,S)
(Sys A,TS)	0	B2	0	B3	B3	B2	B2 **	B2
(Sys A,S)		0				0	B2	0
(Sys B,TS)			0	B3	B3	B3	B3	
(Sys B,S)				0	B3	0	B2	
(Sys B, C)					0			
(Sys C,S)						0	B2	
(Sys C,C)							0	
(Sys D,S)		0				0	B2	0

TS = TOP SECRET, S = SECRET, C = CONFIDENTIAL, ** = CASCADE

Figure 2-4: Military Shortest Path Cost Results

1. Look up the minimum user clearance for the destination security domain's system and determine the larger of the security level of the destination domain and the security level of the minimum user clearance.
2. Use the results of step 1 and the security level of the source security domain as an index into Figure 1-1 to determine the required amount of computer security.
3. From Figure 2-4, look up the actual cost to the penetrator to achieve the information flow between the source and destination domains. If the actual cost to the penetrator is less than the amount of security required by step 2 above, a cascade problem exists.

For the network in Figure 2-1, recall that the minimum user clearance at System A was SECRET, at System B CONFIDENTIAL, at System C CONFIDENTIAL, and at System D SECRET. As an example of the risk acceptance test, consider the flow from (System A, TOP SECRET) to (System C, CONFIDENTIAL). Step 1 of the postprocessing results in a value of CONFIDENTIAL because both the minimum user clearance at System C and the security level of the destination domain are CONFIDENTIAL. Step 2 consults Figure 1-1 using TOP SECRET as the source data sensitivity and CONFIDENTIAL as the minimum user clearance to obtain a recommended computer security rating of B3. In Step 3, referencing Figure 2-4 shows that the actual cost to the penetrator to achieve the flow from (System A, TOP SECRET) to (System C, CONFIDENTIAL) is B2. Since B2 is less than B3, a cascade condition exists for this path. The entry marked ** in Figure 2-4 shows the source and destination security domains that make up the cascade in Figure 2-1. The core cascade path is from (A, TOP SECRET) to (A, SECRET) to (D, SECRET) to (C, SECRET) to (C, CONFIDENTIAL) with a total cost of B2. This path is shown in bold in the upper half of Figure 2-2.

The postprocessing step to test all security domain pairs for a cascade problem can be done in $O(N^2)$ time. There is a total of N^2 domain pairs and the processing for each domain pair requires performing a table lookup from a table of minimum user security levels, from the shortest path results table, and from the table shown in Figure 1-1. The table references plus the comparison of recommended security to the penetrator's actual cost can be done in constant time, resulting in a total $O(N^2)$ complexity for the postprocessing step. Thus the complexity of the overall cascade problem is dominated by the $O(N^3)$ shortest path calculation. Should the actual path causing the cascade be desired, either the all-pairs algorithm in Figure 2-3 should be modified to save the paths as the costs are calculated, or a specific source-destination algorithm should be run on the domain pairs found to have a cascade problem. An algorithm to find the shortest path between a specific pair of nodes is $O(N^2)$ [1], so locating the actual cascading paths can be done without changing the $O(N^3)$ complexity for the overall problem.

2.4. Interpreting the Military Consumption Function

One way to view the military consumption function is that it makes a network risk assessment policy decision that once a computer system with a particular security rating (say B2) is defeated, the defeat of another system with the same level of protection does not cost the penetrator any significant amount of effort. This implies that no matter how many B2 systems are connected in series, a network system created from them will never afford the protection of a B3 system. This consumption function makes sense if one is looking for a worst case analysis of the problem or if it is realistic to assume that the systems in the network suffer from identical or similar flaws so that once one system is defeated, all similar systems are easy to defeat. However, it is easy to postulate other consumption functions based on different assumptions about the (lack of) interdependence of defeating individual systems in a network. For example, assuming that system penetrations are independent events results in a consumption function that is identical to the "normal" shortest path calculation; that is, it minimizes the sum of the edge costs in the path. A corporate cascade example that uses this consumption function is in [4]. As long as the consumption function forms a closed semiring, an $O(N^3)$ algorithm exists for solving the cascade problem under that function.

3. Conclusions

This paper has presented a new method, based on the resource-constrained shortest path, for solving the cascade problem. There are several conclusions:

1. The generalization of the cascade problem and its formulation as a resource-constrained shortest path problem point out the underlying security issues in interconnecting independently evaluated systems; this process leads to a broader understanding of network security risks.
2. Requiring a consumption function to be defined forces a clear policy statement about the (lack of) interdependence of defeating individual systems in a network.
3. A broad set of consumption functions can be defined that allows for a network risk function to reflect a given system's dependence or independence from its peers.
4. The shortest path formulation can detect and locate cascades in $O(N^3)$ time as long as the consumption function and minimization form a closed semiring operating on the graph. The military consumption function presented here and a corporate consumption function in [4] are well-behaved and demonstrate that the semiring requirement is not overly strict.

The resource-constrained shortest path approach and the concept of a consumption function appear quite general. Potential extensions to the basic approach presented here and suggested applications include the following activities:

- Analyze the computational complexity of the algorithm when the consumption function is not as well behaved as in the examples presented here.
- Investigate the effects on the approach when a vector of resources rather than a scalar is involved.
- Explore path-pruning algorithms that incorporate the penetrator resource set into the shortest path calculation step. Compare the path-pruning approach to the method presented here that uses the resource set as a threshold during the postprocessing step.
- Investigate techniques for reducing the number of security domains that must be considered in the cascade problem.
- Compare the ability of the shortest path consumption function to reflect the interdependence of a system from its peers with a statistical analysis of the cascade problem, such as that done by Ted Lee [7].
- Develop precise methods for systems to mutually acknowledge each other's security.

References

1. A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

2. D.E. Bell and L.J. LaPadula. Secure Computer Systems: Mathematical Foundations. Tech. Rept. ESD-TR-73-278, Volume 1, The MITRE Corporation, Bedford, Mass., March, 1973.

3. Dorothy E. Denning. *Cryptography and Data Security.* Addison-Wesley, Reading, Mass., 1982.

4. J.A. Fitch and L.J. Hoffman. A Network Shortest Path Security Model. Tech. Rept. GWU-IIST-90-32, George Washington University, Washington, D.C., September, 1990.

5. John A. Fitch, III. *A Network Security Model Based on the Resource Constrained Shortest Path.* Ph.D. Th., George Washington University, Washington, D.C., 1991. To appear..

6. E. Horowitz and S. Sahni. *Fundamentals of Data Structures in PASCAL.* Computer Science Press, Rockville, Maryland, 1984.

7. Theodore M.P. Lee. Statistical Models of Trust: TCBs vs. People. Proceedings of the IEEE Symposium on Security and Privacy, April, 1989, pp. 10-19.

8. J.K. Millen. The Cascading Problem for Interconnected Networks. Fourth Aerospace Computer Security Applications Conference, December, 1988, pp. 269-273.

9. J.K. Millen. Algorithm for the Cascading Problem. In *Internet IEEE Cipher News Group,* J. P. Anderson, Ed., June 25 IEEE Cipher forum on DOCKMASTER.NCSC.MIL, 1990.

10. Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. CSC-STD-001-83, Department of Defense, Computer Security Center, Fort G.G. Meade, Maryland, August, 1983.

11. National Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. DoD 5200-28.STD, Department of Defense, Fort G.G. Meade, Maryland, December, 1985.

12. National Computer Security Center. Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments. CSC-STD-003-85, National Computer Security Center, Fort G.G. Meade, Maryland, June, 1985.

13. National Computer Security Center. Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements. CSC-STD-004-85, National Computer Security Center, Fort G.G. Meade, Maryland, June, 1985.

14. National Computer Security Center. Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria. NCSC-TG-005, National Computer Security Center, Fort G.G. Meade, Maryland, July, 1987.

15. T.A. Rullo. *Advances in Computer Network Security Management.* Heyden & Sons, Inc., 1980.

16. Rein Turn and Norman Z. Shapiro. Privacy and Security in Databank Systems: Measures of Effectiveness, Costs, and Protector-Intruder Interactions. In *Security and Privacy in Computer Systems,* Lance J. Hoffman, Ed., John Wiley & Sons, Los Angeles, CA, 1973. Originally published as Rand Corporation Report P-4871, 1972.

A CASE STUDY FOR THE APPROACH TO DEVELOPING A MULTILEVEL SECURE COMMAND AND CONTROL INFORMATION SYSTEM

James Obal
Supreme Allied Commander Atlantic
U.S. Naval Base
Norfolk, Virginia 23511-6696

William Grogan
Contel Federal Systems
15000 Conference Center Drive
P.O. Box 10814
Chantilly, Virginia 22021-3808

ABSTRACT

This paper presents a case study of two NATO Command and Control Information Systems (CCIS) projects with stringent computer security requirements. These projects were conceived and initiated at a time when trusted products were not readily available and the concepts of trusted system development and evaluation were not well understood. These circumstances have necessitated the Government and the contractor to seek a unified approach to integrating security into the development process; to ensuring that security requirements are satisfied; and to performing the security evaluation. That approach has been adopted and is now permitting the development of the CCISs to advance. This paper outlines the history of the problems and decisions which culminated in their definition.

INTRODUCTION

This paper provides a description of the lessons learned from the early stages of the two multilevel secure (MLS) CCIS projects. Included are the managerial and engineering decisions which have been taken to help ensure that the project will continue to move forward to completion and satisfy the requirement for B3 certifiability. The importance of demonstrating a sound trusted engineering methodology as well as the role of prototyping in trusted system development is discussed. Particular attention is given to the definition and development of the security documentation which is essential to support both the engineering aspects and the security certification needs of the projects.

BACKGROUND

A fixed price contract to build a high assurance (B3) CCIS for the Supreme Allied Commander Atlantic (SACLANT) was awarded to Contel Federal Systems in October 1984. A second fixed price contract to build a high assurance (B3) CCIS for the

Commander-in- Chief Iberian Atlantic Area (CINCIBERLANT) which has similar functional requirements was also awarded to Contel in October 1987. Both projects are one hundred per cent funded by NATO, and were initially managed independently. Each project had specified unique documentation standards, different engineering design methodologies and separate certification requirements. This duplication of effort soon proved to be extremely expensive and time consuming for both the Government and Contel. The operational needs of both systems were closely analyzed and with concessions being made by all parties the notion of a single system design emerged. Both projects have since adopted a unified security policy, a single set of security requirements, and have been placed under the direction of a single project management office. This joint project is entitled Alpha CCIS, and will be referenced hereinafter as the ACCIS.

The ACCIS is required to process automated messages received from multiple telecommunications lines; to maintain a myriad of databases which contain plain text formatted messages, parametric (record) data, and geographic representations; to provide the capability to create and release formal messages; and to retrieve and display formatted information from its databases.

The ACCIS will combine a suite of alphanumeric and graphical terminals, communication processors, central hosts, and database machines to form the hardware architecture. The system will be highly redundant in order to provide the continuous service requirements mandated by the performance specifications.

Woven into the ACCIS functional requirements is a dominating requirement that the system provide a specified level of computer and communications security. This pervasive requirement for security is principally defined in terms of the so called Orange Book [1]. The ACCIS must be certifiable to class B3 in accordance with the criteria established in the Orange Book. The basic requirement for a B3 system was formed by applying the guidance contained in the Yellow Book series [2]. The B3 requirement is augmented by stringent performance requirements which mandate a high assurance and responsive architecture. The Gemini Multiprocessing Secure Operating System (GEMSOS) developed by Gemini Computers Incorporated was proposed by Contel as the commercial off-the-shelf (COTS) system for the ACCIS Trusted Computing Base (TCB). As another high assurance TCB has subsequently become available (pre-endorsed), the COTS portion of the ACCIS TCB has been re-evaluated. Currently, the HFSI XTS-200 has been identified as the best choice for the ACCIS.

SECURITY POLICY DEVELOPMENT

The ACCIS Security Policy was developed by the Government with the cooperation of Contel. The ACCIS Security Policy contains the administrative, personnel, physical, emanations, communications, and processing security requirements. It is intended to be used both as an operational policy document and as a definition of the requirements for the TCB. The development of the policy was a unique process in that the contractor was reviewing a government originated document for accuracy of content. These reviews were conducted first to determine the consistency, correctness and completeness of the policy and secondly to determine if specific aspects of the policy might pose implementation problems.

The first review process discovered inconsistencies in the policy, mostly due to semantics. However slight, the terminology differences highlighted the need for a glossary of security terms. There were also policy statements that Contel felt were inconsistent with accepted interpretations of the Orange Book but had to remain because of operational needs. An example is the policy's requirement that allows a user to delete a file that is classified at a security level below the user's sign-on security level. There is a potential covert channel associated with this requirement, but it was determined by the Government that the operational need for the feature exceeded the threat of compromise posed by a covert channel.

The second level of review focused on evaluating the impact that implementing the policy would have upon the target TCB. A goal of the ACCIS design philosophy is to produce a system that does not require extensive modifications to the COTS TCB. The selected approach is to layer the additional ACCIS TCB functionality on top of the COTS TCB. The security policy was reviewed with this approach in mind, identifying those requirements that could fundamentally affect the COTS TCB. Additionally, the review produced suggestions for specific policy amendments and recommended design approaches that could be employed to implement the functionality and mitigate the impact on the COTS TCB. An example of a policy impact on the COTS TCB was in the area of auditing. The policy explicitly stated how audit data were to be protected. The method of protecting audit data used by the COTS TCB is equally as strong as the stated policy for audit data protection but it uses a different approach. The policy was modified to allow different approaches for protecting audit data, provided those approaches meet a minimum level of assurance.

Interrelationships between the ACCIS Security Policy, the ACCIS security requirements and TCB design became apparent. A detailed effort to align the ACCIS Security Policy and system security requirements was initiated.

CAPTURING SECURITY REQUIREMENTS

The ACCIS security requirements are a mixture of the standard computer security features specified by the Orange Book (e.g., mandatory access controls, discretionary access controls, identification and authentication, and audit). The ACCIS also has unique security features that are needed to support specific system applications. These features include a Two-Designated-Man-Rule, Trusted Turnover, and trusted message handling functions.

Specific security requirements were identified in the ACCIS Security Policy, the SACLANT Request For Proposals, the CINCIBERLANT Invitation For Bid, and Contel's proposals. Additional ADP security specifications were defined in specific NATO standards and guidelines referenced by these basic requirements' documents. In the process of identifying the security requirements, a basic dichotomy was discovered. The requirement for a B3 system imposed a structure dictated by the evaluation process that did not directly align with the standard systems engineering process. The basis of an Orange Book evaluation is the system's security policy. All certification evidence is derived to some extent from it. This is in contrast to a systems engineering approach which traces the system's development back to the requirements.

To resolve this difference, it was decided that the ACCIS Security Policy would contain all of the security requirements; that is, it would incorporate the security requirements from all of the sources mentioned above. This had the effect of blending the Orange Book evaluation and the systems engineering processes. The

ACCIS Security Policy statements and the ACCIS system security requirements are in complete correspondence.

CERTIFIABILITY, EVALUATION AND OVERSIGHT

Significant emphasis has been placed on the tasks of evaluation, certification and accreditation of the ACCIS. Initially, there existed very broad interpretations of what the term "certifiability" meant to the Government and to Contel. The differences between the evaluation process normally applied to trusted products and an evaluation of an application system that integrated a trusted product had to be sorted out. The issues that arose concerning the question of certifiability illustrate the conflict between developing and evaluating a B3 system and developing a useful system that satisfies its requirements and performs its mission. The initial discussions regarding the nature of the evaluation process revolved around issues concerning the latitude of the evaluators in defining the scope of the evaluation. The prevailing government position was that the ACCIS was a "system" and not a "product" and that a standard National Computer Security Center (NCSC) type evaluation was not sufficient. It was discussed whether the evaluators could mandate additional evaluation requirements regardless of contractual requirements. There were also questions as to how theoretical (vs. pragmatic) the evaluation should be.

The certification evidence document and Contract Data Requirement List (CDRL) item that brought all of these issues to the forefront was the Formal Model of the Security Policy (FMSP). Contel's FMSP was based upon the Bell and LaPadula Model, as are the formal models used by most evaluated TCBs. As with other evaluated TCBs, Contel did not intend to modify the model, but intended to provide an interpretation of how the system mapped into the model. Since other COTS TCB's were also modeled using Bell and LaPadula, this appeared a reasonable approach. However, the Government maintained that the ACCIS was a system and not a product, and Contel was directed to develop a FMSP that was specific to the ACCIS. Accordingly, Contel extensively modified the Bell and LaPadula Model to make it specific to the ACCIS Security Policy. Notions such as ownership and group membership were incorporated. The model's rules were replaced by ACCIS specific rules. The formal proofs were revised but the core theorems, properties and corollaries of the model were preserved, though in a modified form. However, the Government determined that the resultant model, as modified to be ACCIS-specific, was overly complicated and did not effectively represent the ACCIS security policy. Consequently, a joint effort between the Government and CONTEL is in process to rewrite the model without any pre-ordained dependencies on the traditional Bell and LaPadula framework. That effort led to a cooperative FMSP production activity.

Because there were no formal definitions of the certifiability process and the nature and content of the required certification evidence, the resolution of issues like these threatened to stymie the project. As a response, the Government and Contel agreed upon the necessity for a certification plan. To this end, the Government has developed a certification plan specific to the ACCIS. It details the tasks that must be performed by the system evaluators and indirectly, what is expected to be produced by Contel as certification evidence. The certification plan establishes orderings and dependencies between the certification evidence documents. The certification plan is the framework for understanding how the ACCIS can achieve B3 certifiability.

The security evaluator for the ACCIS, entitled Security Certification Technical Agent, is the United States Naval Research Laboratory (NRL). The Certification Authority for the SACLANT CCIS is the United States Commander-in-Chief Atlantic (USCINCLANT) and SACLANT is the accreditation authority. The Certification Authority for CINCIBERLANT is the Portuguese Autoridade Nacional de Seguranca (ANS) and CINCIBERLANT is the Accreditation Authority.

A Security Certification Working Group (SCWG) has been formed to oversee the security aspects of the developing ACCIS system and to reduce the risk of not achieving certification. The SCWG is chartered to provide guidance and resolve security issues as they arise. The membership of the SCWG includes representatives from the ADP security organizations of SACLANT and CINCIBERLANT, NRL, Mitre, Military Committee Communications and Information Systems Security and Evaluation Agency (SECAN), and Contel.

An extensive amount of time and resources will be expended by the Government in order to evaluate, certify, and accredit these systems. The availability of technical support required to evaluate a system of this complexity is very limited and expensive. Careful planning must be exercised to ensure that the contractual milestones and associated deliverables align closely with the certification plan to ensure that these valuable resources are effectively employed and ultimately assist rather than hinder the project's progress.

SECURITY DOCUMENTATION

While there existed substantial guidance on types of security documents required for a B3 system, little information pertaining to document content was available. Consequently, the ACCIS project did not define security-specific Data Item Descriptions (DID) for the security CDRLs. In many cases the only clear documentation specification was that the security documents were to be developed in accordance with the Orange Book. The Orange Book was never intended to be used for defining the content of contract deliverables or for determining the form of certification evidence. Considerable time was expended in SCWG meetings trying to establish agreement on what was expected for each security CDRL. While the security CDRLs closely mirrored the list of certification evidence documentation required by the Orange Book it was not always certain which document would contain specific evidence and if all of the evidence would be accounted for in the complete set of security CDRLs.

To alleviate these documentation problems, the Government and Contel developed DIDs for the ACCIS security CDRLs. A mapping of certification evidence to CDRLs was performed to ensure that all of the evidence would be produced. The NCSC "Guide to Understanding" series, especially the one for Design Documentation [3], was used to develop the DIDs. The Guide for Security-Relevant Acquisitions CDRL and DID Handbook [4] developed by the Headquarters Electronic Security Command, Air Force Computer Center at Kelly Air Force Base, Texas and the Trusted Computer System Security Requirements Guide for DoD Applications [5] developed by MITRE were also used extensively in determining the standards for security CDRLs. Finally, the certification plan was brought into alignment with the evidence mapping and associated DIDs.

Security DIDs had to be written or at least modified to accommodate the Security Policy Input, Formal Model of the Security Policy, Descriptive Top Level

Specification (DTLS), DTLS Implementation Mapping, Security Test and Evaluation Plan, Security Test Procedures, Security Test Descriptions, Security Test Reports, Covert Channel Analysis Report, Trusted Facility Manual and Security Features Users Guide. System documentation follows DOD-STD-2167A DIDs as tailored by the contract.

TCB DEVELOPMENT AND THE SYSTEMS ENGINEERING PROCESS

Both the Government and Contel recognized the importance of not separating the development of the ACCIS TCB and the remainder of the system. Consequently, the TCB development process was integrated into the systems engineering process when these processes were being defined in the Systems Engineering Management Plan (SEMP), the Software Development Plan (SDP), the Software Quality Assurance Plan (SQAP) and the Configuration Management Plan (CMP). Similarly, the security requirements are contained in the Systems Requirements Document (SRD) along with all other requirements.

When unique aspects of the security engineering process required special procedures, those procedures were detailed in the appropriate system plan. For example, the requirements for configuration control of the TCB are more rigorous than for the system as a whole and the CMP describes the additional TCB unique procedures. In the final analysis, requirements for development of the TCB are represented by good engineering practices which were adopted by the systems engineering process.

Contel's ACCIS systems engineering methodology closely adheres to the methodology described in U.S. Army Field Manual 770-78 System Engineering and the U.S. DoD Systems Management College's Systems Engineering Management Guide. The methodology is being augmented by techniques defined in Yourdon's Modern Structured Analysis and in Ward-Mellor's Structured Development for Real Time Systems.

System development will be evaluated at key contractual milestones as defined in and using the criteria of MIL-STD-1521A [6]. Engineering management follows the general guidelines of MIL-STD-1521A, DoD-STD-2167A [7], and MIL-STD-499A [8]. Configuration Management practices and procedures are based upon MIL-STD-480B [9], MIL-STD-483A [10], and NCSC-TG-006-88 [11].

SECURITY TESTING

An early topic at the SCWG meetings was security testing. It was agreed that Contel would perform penetration testing and testing in support of covert channel analysis. The Government could also perform those forms of testing at their option. Less clear was what constituted the testing of the system's security functionality.

The consensus that eventually evolved was that there existed two sets of security functional tests. The first set of tests would be on the system boundary. These tests would be standard, "black box" validation tests, conducted against system security requirements and performed during the factory and site acceptance test phases. These tests would be performed by Contel's testing organization as part of the suite of tests that demonstrate that all system requirements have been satisfied.

The second set of tests would be conducted against the TCB boundary using the DTLS as the basis for the tests. These tests would include the software drivers that exercise the interface into the TCB from the untrusted application environment. Conduct of these tests would be the responsibility of Contel's security engineering organization.

SECURITY RISK REDUCTION

Historically, the development of high assurance computer systems has been troubled by a myriad of technical problems that either delay the project or cause it to be terminated. Both the complexity of the engineering and the complexity of the evaluation processes can contribute to the development problems. In an attempt to identify risk areas early in the development process, the Government devised a demonstration to test the systems engineering process defined by Contel and to assess the complexities of evaluating the product.

To demonstrate their engineering process, Contel was asked to develop a representative "slice" of the TCB, exercising all phases of the engineering process, commencing with requirements definition and carrying the development through to its detailed design. The requirements that were selected involved the processing of messages received from the MLS ACP-127 communications lines. Following the procedures defined in the SEMP, SDP, CMP, and SQAP, Contel produced the TCB slice with all of its required documentation. The results of this exercise significantly aided the refinement of the ACCIS engineering process and provided the Government and Contel with a clear understanding of the complexity associated with developing and evaluating a high assurance system.

TCB design issues are also being addressed early in the development process by prototyping. The purpose of this prototyping effort is to identify solutions to difficult TCB design issues so those solutions can direct the design of the actual ACCIS TCB. This will help reduce the risk that the system may fail to be certifiable at the B3 level. The prototyping effort will be formally documented in the Interface Requirements Specification, Software Requirements Specification and System Design Document CDRLs. The Government and Contel will be able to explore design alternatives, seeking solutions that provide the required functionality without introducing unnecessary COTS TCB modifications or ACCIS TCB complexities. The prototype will also identify early in the design process any operational impacts that may occur by implementing some of the security features as they are currently defined.

The prototyping will include the ACCIS specific security requirements, including the Two-Designated-Man-Rule and Trusted Turnover. The requirement for a trusted data base management system will also be prototyped.

STATUS

Project management for both systems has been delegated to a joint project office located at SACLANT Headquarters, Norfolk, Virginia. In addition to the obvious benefits of centralized management, the logistics support problems inherent in the geographic separation between the European and U.S. sites were removed. The joint ACCIS Security Policy and associated security requirements have been aligned and the development of an ACCIS formal model is in process. A certification plan has been produced and agreed by all parties. The role of the

Security Certification Technical Agent and the Security Engineering Support Agent have been identified and are in place.

Host Nation responsibilities for their respective projects remain autonomous. Contract modifications are in process to unify contract milestones and deliverables. The System Requirements Review phase concluded 10 October 1990. System Design Review is scheduled for November 1991, Critical Design Review is scheduled for October 1992 and Initial Operating Capability is slated for October 1993.

CONCLUSION

The development and evaluation of complex, secure command and control systems is only now being better understood. All of the tools and experience necessary are not yet available. There is a dearth of evaluated high assurance TCBs available to use as a basis for secure systems and there are few in the evaluation pipeline that will ultimately achieve endorsement. Performance requirements can further reduce the number of suitable TCBs for a given system. What should be clear by now is that it simply is not possible at this time to acquire secure systems "off-the-shelf" that satisfy all the requirements of real systems.

The absence of standard security DIDs, certification plans, and secure systems development processes also hampers the ability to define, develop and evaluate secure systems. The Government and the developer must share the same understanding of how security is to be integrated into the system and how that security will be evaluated.

Because the development of secure systems currently involves some uncertainty, strong management support is required. Management must set a course through the sparsely defined territory of secure systems development, identifying deficiencies in the process and finding ways to correct them. A rigorous project management structure is required to ensure that the possibly conflicting interests of building a system that satisfies a critical military mission and maintains a demonstrably high level of security do not bring the development to a standstill.

The ACCIS project can be viewed as a useful case study of the development of a secure, complex military system. It suffered from the lack of tools, products and experience. Fortunately, the mutual desire of both the Government and Contel to complete this project has brought it through the most difficult period.

References

[1] Department of Defense, Trusted Computer Systems Evaluation Criteria, DOD 5200.28 STD, December 1985

[2] Guidance For Applying The Department of Defense Trusted Computer System Evaluation Criteria In Specific Environments, CSC-STD-003-85 and CSC-STD-004-85, 25 June 1985

[3] A Guide To Understanding Design Documentation In Trusted Systems, NCSC-TG-007, Version-1, 2 October 1988

[4] Guide For Security Relevant Acquisitions CDRL and DID Handbook, Volumes 1 and 2, 1 May 1989

[5] Mitre Trusted Computer System Security Requirements Guide for DOD Applications (Draft), 18 March 1988

[6] Technical Review and Audits for System Equipments and Computer Software, MIL-STD-1521A

[7] Department of Defense System Software Development, DOD-2167A

[8] Engineering Management For Total System Development, MIL-STD-499A

[9] Configuration Control, Engineering Changes, Deviations and Waivers, MIL-STD-480B

[10] Configuration Management Practices for System Equipment and Computer Software, MIL-STD-483A

[11] A Guide to Understanding Configuration Management In Trusted Systems, NCSC-TG-006, Version-1, 28 March 1988

CONTRACTORS AND COMPUTER SECURITY - AWARENESS, EDUCATION, AND PERFORMANCE

Ronald G. Brunner
Ronald G. Brunner and Associates
2 Jasmine Court
Rockville, Maryland 20853
(301) 929-1518

Preface

This paper addresses the dual problems of monitoring a contractor's performance and providing adequate computer security within the Federal government environment. Contractors perform many of the government's computer functions, therefore security must be a part of their services and products.

How does the government know that the contractors' are performing the computer security function in an acceptable manner, and that they have the proper level of awareness, commitment, and skills to provide this security? Guidance for determining a contractor's experience, and assuring performance, as they relate to computer security, are contained in this paper.

The paper is intended for use by computer security officers, computer resources management and technical staffs, and contracting officers, as well as by educators who are responsible for training the government personnel. Although the paper is directed toward those individuals within the Federal government, most of what is stated would also be of value to individuals who are working for a commercial organization.

The contents of the paper is based on the author's thirty years of experience working as a computer manager, technician, and educator within both the Federal government and commercial environments.

I. BACKGROUND

A. Contractors in the Federal Computer Environment

During the past thirty years, computer technology, and the way in which in it is used, has changed dramatically. Automated Data Processing (ADP) at one time meant transferring data from written documents to punched cards, for processing by large computer systems in secure data centers. The results of the processing were volumes of printed reports, difficult to handle and use. Any change in the requirements meant days, if not weeks, of work by computer programmer/analysts. Manual activities supported the processing, and when the computer failed, manual work could replace its function. Computers were not easy to use, and they were only critical for a small number of a Federal agency's functions.

Today, terminology has changed. ADP has been replaced by such terms as Management Information Systems (MIS), Information Resources Management (IRM), Federal Information Processing (FIP), and many others which are too numerous to list here. Today data is entered into a computer by optical scanning, voice recognition, remote sensing, data communications, and a variety of other methods. Computers can be large, still housed in a data center, or small, sitting in a person's lap. The data which they process may be shown in printed form, graphically, or be an electronic signal which is sent to a different location to perform another task. Computer programs can be modified easily, sometimes even by non-technical personnel.

The importance of using computer technology has changed also. Today computer technology relates to the sharing of information by the use of local area networks, the creation of documents via word processing, the electronic transfer of funds, the instantaneous location of a unique document, fax transmissions, and a wide variety of tasks and functions. These tasks are critical to the successful completion of an agency's mission. They can no longer be performed manually. The use of computers is no longer optional. This paper will use the term computer resources in the broadest sense, as it relates to all of the technologies and terminologies described above. It relates to any data, information, hardware,software, system, facility, or communications function, where technology is utilized in the performance of an agency's function.

The Federal government is one of the largest users, if not the largest user, of computer technology in the world. It operates thousands of data centers and communications networks, and hundreds of thousands of personal computers. Many of these computer functions are performed by civil service employees, but an increasing number are performed by contractors. Many Federal organizations use contractors to perform the majority of their computer related work, with the civil servants only monitoring the contractors' functions. There are thousands of contracting firms, large and small, whose only source of business is providing computer services to the Federal government. The Office of Management and Budget reports that for FY-1990, the Federal government spent over ten billion dollars for these services. The hardware, software, communications networks, and other computer products which the Federal government uses on a daily basis are also produced by the contractor community. The Federal government, in general, relies completely on the private sector for the computer related products which it requires, and could not function without the products which the private sector provides to it. For FY1990, the Office of Management and Budget reports that nearly five billion dollars were spent on these products.

B. Federal Computer Security Requirements

As computer resources become more of a critical component within the government's work processes, measures have to be taken to assure that these resources are always available for use, for without them, organizations, and even entire government agencies, could stop functioning. Protection has to be provided to guard the government's computer resources from (1) adverse actions such as acts of nature and accidents, (2) improper actions such as malicious and illegal acts, and (3) undesirable occurrences such as system failures due to design limitations and inadequate testing. The integrity, confidentially, and access to all of the government's computer resources must be protected. In this paper, the term computer security is meant to include protection against all threats to all of these resources.

To assure that the computer resources are adequately protected, Congress created the Computer Security Act of 1987, as well as numerous other laws. Federal agencies with oversight responsibility, such as the Office of Management and Budget, and the General Services Administration, have published numerous regulations which all agencies must follow regarding computer security. Individual agencies have created their own rules. Organizations are legally bound to provide security for their computer resources.

The laws and regulations require organizations to be concerned about computer security, and to implement computer security programs. Many government managers have also realized, from a practical viewpoint, that their computer resources are very critical to the performance of their functions, and as a result have taken prudent actions to protect those functions.

C. Contractors' Role in Computer Security

Two major trends within the Federal government have now converged, the use of contractors' products and services to perform the governments' expanding and critical computer related functions, and the expanding concern about the security of the government's computer resources. Contractors therefore must not only be concerned about computer security, they must take an active role in protecting the government's computer resources.

II. THE CURRENT STATE OF COMPUTER SECURITY

A. Contractors' Involvement With Security

Are all contractors fully aware of all of the laws and regulations which apply to computer security, of all of the threats which exist which can adversely affect the government's computer resources, and of the impact to the government if those resources are not available? Most of us will agree that "all" contractors do not have this awareness, and some government employees will state that "many" contractors do not have this awareness.

While this author has not conducted, nor knows of anyone conducting, a formal survey as to the degree of contractor awareness concerning computer security, the author has held dozens of discussions with computer security officers, contracting officers, technical monitors, trainers, and managers within the government, as well as many contractor personnel. These discussions have lead the author to believe that "many" contractors do not have the proper level of security awareness. The fact that you are reading this paper, could be interpreted as indicating that you too are concerned about potential threats to your computer resources, which your contractor is doing little to protect.

Before a contractor can be expected to performance a task, the contractor must be aware of the task. Unfortunately today, "many" contractors are not aware of the need for adequate computer security. They are not aware because their contract with the government does not address it, or because of their lack of understanding about computer security. Computer security is a sleeping giant which is very easy for the contractor, and government, to ignore until a disaster occurs.

Are all contractors fully committed to computer security? The same survey discussed above indicates that there is also a lack of commitment by "many" contractors in implementing a computer security program, or in implementing good security features in their products. This lack of commitment can be the result of a lack of awareness, or it can be that computer security conflicts with the contractors' prime objective, which is to make a profit, or with their client's objectives, which do not include security. If the contract does not spell out what the contractor's responsibilities are regarding computer security, or if the contractor's client is not concerned about security, is the contractor going to do anything about security? Probably not.

Many contractors have the problem of not possessing the necessary skills to implement an adequate computer security program, or to include adequate security features in their products. This is not a problem which is unique to contractors. In general, there appears to be a shortage of computer security individuals who are knowledgeable in both the theoretical and practical aspects of security. This condition is known to anyone who has attempted to recruit an experienced computer security technician.

There are many reasons for the shortage, but they include an ever changing technology, a lack of interest by the computer industry in computer security, and a lack of good, practical, computer security educational programs. Many government agencies and private organizations do offer educational courses on computer security, but the quality of these classes varies greatly, and only a small number of courses are offered. In addition, training funds are usually in short supply. Too often, training on how to develop new systems, install LAN's, or use a state-of-the-art technique takes precedence over security training.

B. Government's Involvement With Computer Security

Before all problems relating to the lack of computer security are blamed on the contractors, the government has to review its own environment. Some organizations within the Federal government have excellent computer security programs, some have adequate security programs, but some just provide "lip service" to computer security. This observation is based on the author's interactions with dozens of government organizations. The lack of awareness and commitment by Federal employees applies to both civil servant managers, computer technicians, and contracting officers.

Many agencies today have serious shortages of civil servant employees. Therefore government employees often state that it is difficult, if not impossible, to monitor on a regular basis what the contractor is doing, as it relates to computer security. Some government contracting officer's technical representatives (COTR) say that they have all to do to assure that the contractor is delivering its products on time, or that it is responding to all of the users' problems, without monitoring what the contractor is doing about computer security.

Even if the COTR has the time to monitor the contractor performance in the computer security area, how do they determine that the contractor's computer security awareness, commitment, and skills are adequate? In too many cases, the COTR does not know enough about computer security to question the contractor about it.

C. Contracts

What work a contractor does, or does not, do is defined in the contract which exists between the government and the contractor. The contractor is not going to do anything not contained in the contract because the contractor will not get paid for it. In some cases, if the contractor performs work not contained in the contract, it could even be considered as being an illegal act.

Computer security has not been adequately addressed in many of the government contracts the author has reviewed. Many times this is because the technical security requirements have not been determined, other times because the necessary security clauses have not been kept current, and still other times, because there are no funds available to include any security features.

III. IMPROVING COMPUTER SECURITY

A. Overview

You are very concerned about the security of your computer resources. You do not believe that your contractor is doing an adequate job in protecting those resources, resources which must be operational for your organization to fulfill its mission. How do you get the contractor to be more responsive and to protect the computer resources, and maybe even your job? Perhaps, you do not even know whether your contractor is performing in a satisfactory manner or not, or whether the contractor has the skills and motivation to do the job. Or maybe, both you and your contractor know exactly what should be done, but you can not convince your management to authorize the necessary security program, or there are no funds to perform the work. The remainder of this paper will provide guidance to anyone who is concerned about these conditions, or has to educate government personnel, such as contractor monitors, about computer security.

B. Government's Awareness

The first thing any government organization must do to develop and implement an adequate computer security program, is to make the government's managers and technical staff aware of the legal and management needs for computer security, and to educate them as to what computer security really means. You can not place requirements on your contractor, if you and your management do not understand the requirements, or if there are no funds to perform the work. You can not tell your contractor that computer security is important, if you and/or your management do not agree. You can not monitor a contractor's technical performance if you do not understand what the results of that performance should be.

Awareness and education of the government personnel concerning computer security are necessary before you can get the contractor involved. A contractor who is knowledgeable in the area of computer security, can assist you in "selling" security to your management, and in the education of your staff regarding the technical aspects of computer security. The contractor can not be the driving force behind computer security, it must be the government. Step one then in any computer security program is to assure that you, your management, and technical staff are knowledgeable about the legal, management, and technical requirements for computer security, and are committed to a reasonable and adequate security program.

How do you accomplish that? Lengthy papers have been devoted to informing and educating people about computer security, and to completely address the issue here would not be practical. Briefly though, the legal and management need for computer security must be made known to all. It is the process of awareness. You have to know that if you do not have an adequate computer security program, you are not complying with the law. You have to be aware that your organization may be placed at great risk if you do not have an adequate computer security program. Your computer resources could be vulnerable to a wide variety of threats, which could have a significant adverse effect on the mission of your organization.

Those threats not only include computer viruses and white collar criminals, but also the results of unusual weather conditions, the failure of a sprinkler system, a labor dispute, the detection of a hazardous material in your physical environment, a design error in your computer system, a feature in a product which does not work as promised, and thousands of other actions and events which occur in computer environments throughout the world on a regular basis.

After awareness has been established, a program of computer security education must take place within the government. The government does not have to understand the details of how a specific virus works, or what are the different types of data encryption. or how an uninterruptible power system functions. They do have to be able to recognize what are the potential threats to their resources, where and how they are vulnerable to those threats, what will be the effect on their operations if the threat turns into an adverse action, what technology and management practices exist to counter those vulnerabilities and adverse actions, and whether the cost for this protection can be justified. The question must be addressed as to how much computer security, and at what cost, is appropriate. This is risk management. A contractor can assist you in gathering all of the required information and performing the necessary analyses, but the final decision as to which risk is not acceptable and therefore must be protected against, and which risk is acceptable, for which no protection will be provided, must be made by the government.

C. Expanding the Contract

After it has been determined what level of computer security is required and must be provided by the contractor, it must be addressed in a contract. If your computer security requirements are not described in your solicitation, and not detailed in the resulting contract, those requirements are not going to be satisfied by your contractor. You can not assume that the security features or commitment you desire are going to be provided, nor that the contractor will want to, or be able to, provide features or work not contained in the contract. After a contract has been created, it can be modified to add security requirements to it, but that is usually costly, sometimes difficult, and always politically a problem. You must ask your boss for more money than you had planned for, and he/she asks why?

Your security requirements must be described in detail in your solicitation. Any specific security requirements which are unique to your environment must be inserted. General requirements can be obtained from your agency's procurement "boiler plate", or from prior solicitations, but usually that will not provide all of the protection you need.

Your solicitation must address your overall needs, and what security features and services those needs require. Areas to be included are: how is security controlled by the hardware and software; what security features do you require in the products and systems; how is access to the resources controlled; what user and technical documentation is needed; what are the legal considerations; how are communications to be protected; what is the importance, confidentially, and criticality of your data; what clearances and skills must the personnel possess; how will security training and awareness be handled; what are the security needs for the facilities; and how will contract administration be conducted. Assistance in placing the proper requirements in your solicitation is provided by the National Institute of Standards and Technology, as well as the General Services Administration.

In addition to your security requirements, you must describe your environment to your contractor. If a risk analysis just determined that your installation has serious security problems, you must tell the contractor. If you do not, the contractor may not address those issues, and costs, in the proposal. If a contingency plan exists to be used in the event of a computer failure, the contractor must know about the plan. Anything which can affect the security and the functioning of your computer resources must be told during the solicitation phase to your potential contractors.

Many times, all of the security requirements which you desire the contractor to respond to, can not be identified at contract award, or if they have been identified, there are no funds to support them. It is therefore wise to include in any contract, options for the contractor to perform additional security activities, if the requirement and/or funds arise. Options do not commit the government to having the contractor perform the work, but they can save significant time and procurement effort if the government does require the work to be performed.

D. Responsibilities

The solicitation, and the resulting contract, must make it clear as to the responsibilities of the government and the contractor. Who trains the contractor, pays for the training, determines the level of training required, and determines when it has been successfully completed? Who creates the risk assessment, screens the contractor's personnel, and controls the passwords to access the computer system? The who, and how, and what, and when, for all activities must be included. The specific areas to be considered included: computer security planning, risk determination and analysis, identification of sensitive systems, contingency plans, training, procurement of additional security related products and services, personnel requirements, determination of costs and available funding, and controlling and monitoring the contractor's performance.

Unless these responsibilities are clearly defined, several adverse actions will occur. First, the government and the contractor will be arguing throughout the term of the contract as to who is doing what and who is going to pay for it, and second, the even more important, required functions may not be performed.

General guidance is that the government is responsible for overall planning and control, conducting awareness training, the identification of sensitive systems, and funding. The contractor would perform all of the required analyses, do the detailed training, and be responsible for the day by day security actions. The exact breakdown of responsibilities will vary from contract to contract. It is very important that all responsibilities be identified, and assigned to either the government, or the contractor, or even a different contractor.

E. Determining The Contractors Awareness, Commitment, and Skills

You have defined in your contract what support you require, but how do you know if the contractor has the capabilities to provide that support? The contractor's proposal states that they have the knowledge and experience, or their product performs that function, but how can you be sure? Techniques which are used include: (1) reviewing their past performance, (2) assessing their plans for future performance, (3) interviewing their proposed personnel, and (4) seeing a demonstration of their proposed products.

Past performance can be determined by checking with organizations who the contractor has supported in the past. The contractor says that they have performed these security activities for this client. Check with the client to determine is that true, and whether the client was fully satisfied with the contractor's performance. Check with many prior clients, and with both the technical and procurement monitors. Determine if the contractor have the skills, commitment, and record for doing what they say they can do. If the contractor is proposing personnel, check the references on the resumes.

In addition to past performance, you want to know what the contractor plans to do for you. Require the contractor, in their proposal, or response to your task order, to provide to you a detailed plan as to how the security work will be accomplished. The plan should address exactly what is going to be accomplished and how. What is the level of staffing proposed, and what are the qualifications of the staff? What is the detailed schedule with interim milestones? What are the responsibilities of the contractor and the government? What are the deliverables, and how will the government know that the work has been performed in a satisfactory manner? This plan must be agreed to by the government before the contract is awarded, or the task signed off by the government. It should provide to the government a feeling that the contractor understands the security problem, has the talent to attack the problem, and possesses a plan to resolve the problem.

A part of a government contract, should be the contractor's security training plan. The difficult question is, what is the correct level of training? There is no correct answer, for the answer will depend on the required level of security, the funding which is available, the sensitivity of the application, and many other factors. You should assure though that the training proposed corresponds to all of those factors.

At the time a contractor submits a proposal to the government, it is proper and advantageous for the government to interview some, if not all, of the contractor's proposed personnel. This will assure that they are committed to working on your project, and that they possess the necessary skills to support your security program. If you do not have the necessary skills to adequately interview them, obtain those skills from elsewhere in your agency, or even hire another contractor to assist you. Do not assume that because the proposed resume says that the person has the required experience, that the person actually does have the experience.

Benchmarks, or product demonstrations, can be used to prove to you that the proposed product or technique, actually accomplishes what it is suppose to accomplish. This "hands on" viewing of what is being proposed can be very detailed, and require many hours of time by both the government and contractor personnel, or it can be just a brief demonstration at another client's site. Some contractor's products are formally approved, or accepted, by some government agencies, or the commercial marketplace. This approval or acceptance could be used instead of having your own demonstration, but be sure that the environment in which the product was approved or accepted is exactly the same as yours. Demonstrations do not only apply only to products, they also apply to techniques. If the contractor proposes to conduct their own in-house training program, it is appropriate for government personnel to sit in on those training sessions to assure that they will meet the government's specific requirements. If the contractor is to develop a contingency plan, the government may review prior contingency plans the contractor has created to assure that the approach is acceptable. Do not accept promises from the contractor. Rather, view with your own eyes what you are going to receive from the contractor, and determine whether it meets the contractual requirements.

F. Monitoring Performance

You have now been convinced, at least in theory, that your contractor has the skills, experience, commitment, and plan to protect your computer resources. How you do know that the contractor is doing what the contractor has promised to do. How do you monitor and control the contractor? Especially when your staff tells you that they do not have the time to perform this monitoring function. First, the question of priorities must be addressed. If computer security is really important to your organization, and if you really desire to control what your contractor is doing, you will have to locate resources to monitor the contract. They do not have to be full time resources, but they do have to be at a level which relates directly with the importance of what the contractor is doing.

The best way to monitor your contractor, at any level, is by using multiple control points. What is meant by that, is that you obtain, on a regular basis, data concerning your contractors performance not from one source, but from multiple sources. You then compare all of the data to assure that it is consistent. If it is not, you have a problem. For example, you receive from your contractor status reports telling you of the good things the contractor is doing for you. At the same frequency, you should also receive similar data from your user community, your operations personnel, even from other contractors concerning the contractors performance. Is all of the information consistent? You receive written reports concerning your contractors performance. Does verbal inquiries agree with the written data? In walking around the entire environment being protected, does your visual inspection and discussions agree with the written data? If it does not compare, you better start asking many more questions.

For example, how do you know that the contractor's employees are receiving the security training which was proposed. First, obtain from the contractor detailed information concerning which contractor employees went to class. Since the government is paying, either directly or indirectly, for the training this is an appropriate request. Contact the trainer yourself to obtain feedback as to who was trained, and what was their performance in the class. Talk to government employees who may have been in the same class. Submit to the contractor a simple task concerning the material which was covered in the class, to be completed by the students who just completed the class. Have the results of the task reviewed by competent security sources. Does all of the information you have gathered agree, or not?

Remember though, that any information, especially written, which you require from the contractor should have been identified in the contract. This does not mean that every report has to be listed, but categories and frequencies of reports should be addressed. You can only perform the contact monitoring specified in your contract, and/or permitted by government laws and regulations.

G. Contract Administration

Most interactions between you and contractors are to be in writing, and flow through your contracting officer, or at least the contracting officer's technical representative. Too often that does not occur. Verbal direction, usually illegal, is provided to the contractor by a variety of government employees. The result is confusion on the part of the contractor because they do not know who to respond to, frustration by the government's technical staff because the required work is not being performed, and anger by the procurement officer because the laws and contract are not being followed. The government's procurement laws and regulations must be followed in administrating any contract, not just because they are the laws, but because good management practices require that they be followed. In addition, there must be a good contract "audit trail", that is a file of documents showing what the contractor was to do and what they actually accomplished. Letters of commendation as well as complaints must be included. Do not assume that the contractor will respond to your verbal requests or concerns. Put it in writing, send a copy to the contracting officer, and place it in the file.

Either because you are doing a good job in monitoring your contractor, or because a disaster just occurred, you determine that your contractor is not accomplishing what you thought the contractor was accomplishing. What do you do now? First, you have to determine if the function is addressed in the contact or not. If it is not in the contact, you can not force the contractor to do something which the contractor is not legally bound to do. If your legal and procurement staff tell you the contact is not clear, "the monkey is on your back". Your only real course of action, is to have the contracting officer issue a modification to the contract, or to obtain other resources to get the job done. Hopefully you will have learned from your mistake, and will write a better solicitation, or task order, the next time.

Suppose though, the contract is complete and clear, and your contractor is just not performing. Initiating the disputes, default, and termination clauses in the contract is normally not the way to go. The contractor, your contracting officer, and your management, will all get mad at you, but the work still is not getting done. Instead attempt to determine what is the problem. Most contractor problems have been caused by incorrect, incomplete, or erroneous communications. Therefore, when a problem does arise, talk to your contractor project manager, vice president, or the owner of the company, and determine what the problem is and how it can be corrected. The contractor usually is just as interested as you are in solving the problem. Contractors can obtain additional contracts only if their prior performance has been acceptable. The last thing they desire is to have an unhappy client.

It is possible though that you can encounter the contractor that can not or will not perform. The task then is one of terminating the contract, and obtaining a new contractor. Remember, that if your contract is clear and complete, and you have good administration records, there should be no question concerning the contractor's lack of performance. These documents will permit you to terminate the contract in a shorter time, and with less frustration, than if things are not documented.

You have placed the security of your computer resources in the hands of your contractor, but computer security is still the government's responsibility. You must work together with your contractor to attack and resolve your security concerns. In this way, the resolution of most problems will occur in the shortest of time, the protection of the computer resources will be maximized, and everyone will benefit. Computer security is a sleeping giant. You are going to need all of the help you can get, to properly protect all of your computer resources, from those bad things, which are guaranteed to happen to you.

COVERT CHANNEL ANALYSIS PLANNING FOR LARGE SYSTEMS

Lee Badger
Trusted Information Systems, Inc.
3060 Washington Road (Rt. 97)
Glenwood, MD 21738

Abstract

Covert channel analysis is a challenging task, particularly when performed during the development of a large system. Some elements of covert channel analysis, such as timing channel identification and reduction, require techniques currently beyond the state of the art. Performing a useful covert channel analysis during development requires a careful balancing of costs and assurance, and a careful selection of currently available techniques. While it is possible for new research to assist in the covert channel analysis of large systems, developers cannot plan on breakthroughs. This paper discusses available techniques, their limitations and tradeoffs, and makes recommendations for performing covert channel analysis.[1]
Keywords: Assurance and Analytic Techniques, Conducting Security Evaluations.

Introduction

Covert channel analysis (CCA) is a process of identifying and analysing information flows in a security policy model, system specification, or system implementation. CCA is required to satisfy the TCSEC [1] B2 and higher evaluation class requirements and also the ITSEC [9] E4 and higher assurance levels. CCA may be performed either informally or formally. In general, CCA has 3 distinct components: 1) identification of covert channels, 2) estimation of their capacities, and 3) reduction of capacities. An additional, implicit component of CCA, is to gain assurance that each of the three tasks are correctly performed.

Performing a credible CCA, successfully and at reasonable cost, during a large (i.e., complex) system's development is a challenging task. In the context of the Trusted Mach system currently under development at Trusted Information Systems, a CCA plan has been evolved that balances concerns over assurance, cost, and feasibility. This paper first provides definitions and summarises available techniques. It then compares the techniques, and presents an approach for performing CCA during system development.

Definitions

Generally, covert channels make use of system characteristics, such as error return codes or global identifiers, that are not normally thought of as containers of information but that reveal the state of shared resources (hence the word "covert"). In contrast, information flows that occur between system "objects" as a result of using system primitives in the intended way can normally be thought of as "overt channels." A *covert channel* is usually defined to be a "communications channel that allows a process to transfer information in a manner that violates the system's security policy."[1] A system security policy (for B2 and greater systems) should be completely stated in its FSPM (Formal Security Policy Model[1]).[2] If the FSPM includes an information flow policy, such as noninterference [6] or nondeducibility [18], this definition is accurate. For access control FSPMs (e.g., [3]), however, the system security policy makes no statement about information

[1] **This work was supported by DARPA/ISTO Contract MDA97-90-C-0027.**

[2] **The construction of an FSPM that accurately reflects external security requirements is beyond the scope of this paper; a valid FSPM is assumed.**

flow, and the "intent" of the system security policy must be inferred from the properties of the defined secure state. For example, the ss-property and *-property of the Bell and LaPadula FSPM imply an information flow rule of "no flow down." For these systems, the purpose of CCA is to gain assurance that information may not flow contrary to the *intent* of the system's security policy. It should be noted that this definition is *very* broad, including as covert channels all mechanisms that reveal failures by the TCB (Trusted Computing Base) to satisfy the FSPM.

Although not required by the definition of a covert channel, the general paradigm of covert channel exploitation involves one or more sending subjects that have access to sensitive information, and one or more receiving subjects that have lower access to sensitive information. Under the assumption that components of the TCB do not intentionally compromise information, there must at least exist a receiver that is untrusted. If there is no untrusted sender, the receiver's actions amount to spying on events going on in the TCB, which satisfies the definition above, but is a much smaller threat because there can be no cooperation between sender and receiver, and because presumably the TCB is using care in its handling of information. Most of the literature on covert channels focuses on the more dangerous case, where both sender and receiver are untrusted subjects and cooperate. In this case, the sending subject must be executing a trojan horse program that is using the access rights of a highly cleared user. Although there may be many senders and receivers to exploit a given channel, the number is an implementation detail of the exploitation; this paper refers to "the sender" and "the receiver."

Typically (and in the TCSEC), covert channels are divided into two classes: storage channels and timing channels. A *storage channel* is a covert channel in which the transmission of information involves the alteration and observation of storage locations in the TCB. A *timing channel* is a covert channel in which the transmission of information involves the manipulation, by the sender, of the length of time that the receiver requires to perform some operation. For a timing channel to exist, the receiver must have access to a timing reference in order to measure the time required. Some channels are difficult to categorise as either timing or storage[23]. For example, the following channel would appear to satisfy both definitions: a sender positions a disk's arm to the middle or outer track of a disk by performing I/O to files that are known to reside in those places; the receiver performs I/O to an inner track, measuring the delay in servicing the I/O request. The position of the arm is internal state (storage), and the receiver deduces that information using timing properties.

Because covert channel exploitations bring about the disclosure of information, a definition of information is necessary. Although the intuitive definition of information as "bits" is useful, a more formal foundation is required to calculate the capacity of a channel, that is, the rate at which information flows through it. Shannon's definition [17] is widely accepted as the proper foundation. Very informally stated, information is the amount of "surprise" that the receiver experiences when learning the value of a symbol received. As an example, a receiver that receives one of n symbols (all equally likely) learns more than a receiver that receives one of m symbols (all equally likely) when $n > m$. The amount of information received depends on the probabilities of the symbols. If one of the n symbols is very likely, so that the rest are very unlikely, then receiving one of the rest is relatively "surprising," and more information is received than would be the case if the probabilities were equal. Because the overhead of sending different symbols may vary dramatically, covert channel exploitations may substantially increase channel capacity through the use of coding. Using coding, a sender can change the probability distribution of symbols received by the receiver by encoding expensive symbols as sequences of less expensive symbols.

TCSEC B3 Requirements

The TCSEC requires a system developer to conduct a thorough search for covert channels (storage channels only at B2; timing channels also at B3 and A1), and to determine channel capacities for identified channels using either actual measurement or engineering estimation. In the recent Trusted Xenix B2 evaluation, the evaluation team rejected the use of actual measurement because there could be no guarantee that the strategies and code used to drive the channel were the most efficient possible. Analytic techniques must therefore be used for measurement (note: just as measurement is prone to underestimation, analytic techniques are prone to overestimation). The TCSEC criteria refers the reader to the covert channel guideline section of the TCSEC for guidance on both acceptable capacity and auditing. The guideline asserts that all channels with capacities above 1 bit per second can be audited without adversely affecting system performance, and therefore that such channels should be audited.[3] Additionally, it recommends auditing of channels with capacities above 1/10 bits per second where possible. Since the guideline is not part of the criteria, however, it is subject to modification by precedent. During the Trusted Xenix B2 evaluation, the team found the following channel capacity and auditing categorisation acceptable:

Capacity	*Action*
< 1	no concern
$1 - 10$	document, audit if possible
$10 - 100$	if not possible to reduce, audit and document
> 100	not generally acceptable

ITSEC Requirements

Development of ITSEC rated systems has 4 phases: requirements, architectural design, detailed design, and implementation. At the E4 assurance level, CCA is required in the detailed design phase. At E5 and E6, CCA is required both during the detailed design and implementation phases.

In the detailed design phase, a specification using "some form of rigorous approach and notation" is required. The specification is required to provide a DTLS and to identify all security mechanisms. A "design vulnerability analysis" must be conducted on the specification to determine how security may be subverted on a system configured in a specific way by a security administrator. This analysis must identify covert channels. It is required that the exploitation of covert channels be auditable. In the implementation phase, an "implementation vulnerability analysis," for a given configuration, must identify covert channels.

The ITSEC targets covert channel analysis at specific configurations, which opens the possibility of supporting both covert channel analysed configurations and other, perhaps more useful, configurations. In general, the ITSEC does not appear mature in its treatment of covert channels. First, a definition is not given, although the reader might be justified in using the TCSEC definition. Second, the requirement that all channels be auditable is probably not technically feasible.

Channel Identification

There is currently no known technique for identifying all covert channels in an implementation. Relatively high confidence can be gained that storage channels have been eliminated from specifications [11] Unfortunately, implementation details not present in an interface specification may introduce new channels. For a complete, rigorous treatment of storage channels, it is probably necessary to combine analysis of interface specifications with code level (or very low level specification) checking to validate the interface analysis. Although at least one informal methodology exists for searching for timing channels (summarised below), identification of timing channels remains ad hoc. Most of these methods focus on finding "potential" channels; informal techniques must then be used to determine if the channels can actually be used. This section presents three different approaches to finding storage channels, and one approach for finding timing channels.

Shared Resource Matrix Methodology

The shared resource matrix (SRM) methodology of Kemmerer [10] focuses on identifying the shared resources whose "attributes" can be used for covert channel exploitation, and the system primitives that must be used to manipulate the attributes. As defined by Kemmerer, a shared resource is "any object or collection of objects that may be referenced or modified by more than one process"[4]. The definition of the storage

[3] The auditing of some timing channels, if attempted, would severely degrade system performance.
[4] Kemmerer assumes that subjects are processes.

objects in the system's security policy model determines a subset of objects about which attributes may exist for covert manipulation. For example, a file or terminal may be a shared resource that has a sise or lock attribute that is subject to manipulation. Subjects may communicate by changing the sise or setting the lock. Whenever multiple subjects share a cpu, an additional shared attribute, the response time, is also available.

Kemmerer gives the following minimum criteria for the presence of a storage channel:

1. Sending and receiving subjects have access to an attribute.
2. The sender can cause the attribute to change.
3. The receiver can detect the change.
4. The sender and receiver are able to synchronise.

Timing channels have a slightly different set of criteria:

1. Sending and receiving subjects have access to an attribute.
2. The receiver has access to a time reference.[5]
3. The sender can modulate the receivers response time for detecting a change in the attribute.
4. The sender and receiver are able to synchronise. [6]

The methodology is applied by first identifying the shared resources and their attributes, and then the system primitives that can be used to manipulate them. This information is then organised into a matrix where the rows correspond to shared attributes and the columns correspond to available primitives. The elements of the matrix are labeled by a R, M or both to indicate whether the primitive observes the attribute, modifies it, or both. If the row for an attribute contains both R and M, it may be usable as a covert channel. A weakness in the methodology is that it does not state how to identify the attributes or primitives, or how to determine whether or not an exploitation is possible. Due to this informality, concluding the analysis is a subjective decision. At the lowest level, analysis is performed on every primitive: 1) that a subject may invoke, and 2) that causes or observes a system state change. These primitives include all system traps, kernel interface calls (which are interpretations of system traps), functions made available by trusted processes, and cpu instructions. Depending on its arguments, for example, the move instruction may affect system memory or cache contents; these effects may be visible to subjects at other security levels.

Once constructed, the matrix is transformed by calculating the transitive closure of the information flows. The transitive closure simply extends all direct information flows to include indirect flows as well. Both Tsai [20] and Levin [11] have argued that this step is not necessary because indirect flows must be based on direct flows. Levin notes that, because an exploitation may exist for indirect flows, such a conclusion is only justified if no direct flows are eliminated from consideration as having no exploitation. Some tools exist for assisting in constructing an SRM from specifications. Gemsos [11] used FDM (Formal Development Methodology) tools [5] to generate a SRM for storage channel analysis. The system interface was described using the Ina Jo specification language.

[5] Actually, Kemmerer asserts that both sender and receiver need access to a time reference. It does not appear necessary, however, for the sender to have such access so long as the actions that the sender takes are known in advance to affect the receiver's response time.

[6] This is not necessarily a significant requirement. In the absence of synchronisation, variations in the sender's and receiver's relative speeds show up as noise in the transmission (which can be eliminated using suitable encoding). Synchronisation is required for a noiseless channel, however.

Noninterference

A subject is "noninterfering" with another if the subject's actions do not affect the other subjects view of the system's behavior [6]. If we view a system as a sequence of inputs and outputs, noninterference can be stated: subject A does not interfere with subject B if, for every sequence *w* of inputs and outputs of A and B, the output seen by B is identical to that which would be seen by B in the sequence that is identical to *w* except that all of A's inputs have been deleted. A system has the MLS property if, for every subject A whose security level properly dominates that of another subject B, A does not interfere with B. These ideas can be more precisely stated; a state machine definition is given in [6].

Noninterference is characterised by system behavior at an interface; the interface may be at any level of abstraction. Noninterference belongs to the family of information flow models because the satisfaction of the policy can be shown by demonstrating that no information flows between noninterfering subjects (as defined by the label dominance relation). Because covert channels are means by which high subjects can interfere with low subjects, a system that has the MLS property has no covert storage channels.[7] Covert channels may then be discovered by attempting to show the MLS property for a system, and examining the places where the proof fails. This approach was used, in comparison with the SRM methodology, to analyse specifications of the Secure Ada Target for covert storage channels [7].

The noninterference approach has the advantage that, unlike the Shared Resource Matrix approach, it is possible to "know when you are done." The method of analysis, however, is extremely arduous. Constructing proofs that source code satisfies a particular specification is extremely difficult; producing arguments about where and why a proof fails (and that the proof could not in fact have succeeded) is even more difficult.

A Code Level Technique

Although the SRM methodology [10] provides an approach to identifying covert channels, it leaves out the specifics of how to find channels in source code. Tsai [19] provides a way to identify channels in C source code using semi-automatic analysis. It is claimed that the method is formal and that all storage channels are found. Although the method does use some formal techniques, the strength of the results is limited by the strength of the (to date, informal) correctness claims for system implementation in general. Additionally, the choice of C as the implementation language makes the analysis vulnerable to incorrectly implemented pointer manipulations that cannot be caught by the analysis. Tsai's method can be seen as an extension of the SRM methodology. It can be described in three broad phases:

Identify trusted interface primitives: This information is available from the system DTLS.

Determine the visibility/alterability of internal TCB variables This determination starts by first examining, using dataflow concepts, whether or not variable values are (potentially) returned to a caller of a TCB primitive, or are potentially altered by call. For example, the statement "x = y;" causes information to flow from y to x. If the statement is guarded by an "if B", then information flows from B to x as well. Dataflow rules for tracking information flow in code have been given by Denning [4]. This analysis is performed on a function by function basis. Potential function call paths are then examined by discovering which functions can be called from each TCB primitive. TCB variables that can be set or observed from the TCB interface are then flagged as covert channel attributes for a code level SRM. The TCB primitives from which the attributes can be modified or from which the attributes are visible are identified as the columns of the code level SRM.

Analyse shared attributes (and weed most out): The criteria for weeding out identified attributes are not formalised. Attributes may be weeded out either because the information flows supported are legal, or because they confer no useful information.

Tsai's method identifies numerous attributes. It does not, however, provide a formal way to determine which

[7]In [7] it is stated that a system having the MLS property might still have timing channels, because there is no explicit representation of time in the noninterference model.

are actually harmful. Ideally, an FSPM would be mapped down onto the code, and the legal channels could at least then be formally eliminated. The "no useful information" channels are more difficult yet.

A weakness of the method is its focus on global variables. First, such variables should be considered in *assembler* as well, and also i/o should be analysed. Additionally, it should be possible to include the CPU instructions as part of the TCB interface. I/o can present obvious channels, such as the print job identifier channel for Unix. In that channel, print job numbers are written into a file in a DAC-protected directory by the trusted printer daemon. New jobs are numbered after the last job written into the file. Because users cannot access the directory directly, they cannot read the file, but they can notice which job numbers are assigned to their print jobs. Using the SRM methodology, such channels can be detected by identifying the resource consisting of print job numbers.

Tsai's method, used in Trusted Xenix [2], identifies essentially three kinds of storage channels:

resource exhaustion A resource pool (e.g., a memory allocator) returns an error message when there is no more resource to allocate.

policy conflict An operation that may compromise information, but which must be maintained for compatibility or usability. For example, some systems refuse to remove directories when their (high) contents are not empty.

event count Channels in which the sender can manipulate a (usually integer valued) index or size attribute of a resource. For example, a report of the total number of free disk blocks is an event count channel.

Timing Channels

Timing channel identification has historically been ad hoc. In order to measure the time that an operation requires, the receiver of a timing channel must have a point of comparison. The most obvious such point of comparison is the system clock. Points of comparison need not be so obvious, however. As stated in [22], a timing channel may exist whenever there are two or more clocks where a clock is defined to be "any series of events, visible to a process, which may be used by the process to measure the passage of time."

In [22], Wray proposes a methodology that focuses on the identification of clocks. Using the methodology, all clocks are identified and an N by N matrix for the N clocks is constructed. The vertical axis would list the clocks to be modulated by the sender, and the horizontal axis would list the clocks to be used by the receiver to measure the modulations. Except for the diagonal of the matrix, each cell can be filled in with the modulation scenario. It is not possible to modulate some clocks. This technique is not extremely different from the SRM approach. Clocks are discovered by first listing "clock classes" (an informal activity), and then subdividing the clock classes by their internal events. For example, some clock classes proposed in [22] were: instruction timings, operating system calls, the system clock, and disk I/O transfer time.

Once clock classes have been identified, individual clocks (usually subparts of a class, for example the different interrupts for a disk transfer) are identified, and example exploitation scenarios are hypothesised. For a particular pair of clocks there may be a large number of possible exploitation scenarios. Choosing the fastest and most difficult-to-audit scenarios is an ad hoc process. In [22], Wray provides a number of example exploitation scenarios:

disk-arm The sender positions the disk head by performing i/o on known tracks. The sender issues two read requests (to different sectors) and examines the completion time of two read requests.

disk-arm write Similar to the above, the sender first positions the disk head. The receiver issues two write requests such that they partially overlap on the disk and such that one will happen first depending on the position of the disk head. The location of the disk head is revealed by which value remains. This is an example of a "direct" channel, in which the information is deposited on a medium without the receiver learning it first.

printer write with timing loop The receiver issues print requests and waits in a timing loop, after which it cancels the request. The sender modulates the length of the receiver's timing loop by contending for memory access.

bus contention A high processor modulates memory access contention with low processors. This channel is potentially large.

cache reload The receiver fills the processor cache with low information. The sender causes some cache entries to be invalidated, and the receiver then notices the time delay in accessing memory.

Other timing channels have been presented in the literature; an exhaustive list of reported channels is beyond the scope of this document.

Channel Capacity Estimation

Channel capacity estimation should be done after the identification phase is complete. The capacity estimates serve as input to the channel reduction process. Capacity estimation has three components:

- measuring the time each TCB primitive requires to execute,
- finding scenarios for the manipulation of each channel, abstracting the scenarios to gain a guarantee that no other scenario exists that can drive the channel at higher speeds, and
- estimating the rate at which information can be transmitted using the abstracted scenario.

In principle this approach works for both storage and timing channels, but techniques for finding the information rates of abstract scenarios may differ. This section discusses each of these components.

Measurement

Measurement requires that, for each evaluated hardware base, all TCB primitives, that have been related to covert channels in the identification phase, be timed. Both kernel and server interactions will require timings since CCA will be performed for both the kernel and servers. It can be difficult to obtain believable timings for TCB primitives. Primitives may execute much faster than the clock ticks of the system clock used to measure the time.[8] In this case, it is necessary to time n calls of a primitive. For primitives that allocate (or deallocate) resources, however, it may not be possible to execute the primitive n consecutive times. Primitives may have to be paired (allocating and deallocating) to measure their composite timings. Many primitives may require different amounts of time to execute depending on the system state. Characterising the state is sometimes possible (e.g., file creation in a large directory is slow), but often the state is such a complex result of previous system history that analysis is not feasible. To blend the differences, the timings should be measured multiple times, and confidence intervals should be used to gain assurance that actual times are close to measured times with high probability.

Many primitives transmit information through failure conditions; it is therefore necessary to measure both calls that succeed and calls that fail. An additional, unquantifiable, concern is that the time that a primitive requires to execute often depends on what arguments are provided it. Arguments can sometimes be selected that "do no work" (resulting in fast executions), but covert channels cannot in general be driven by "null" operations. "Reasonable" arguments must be chosen.

Idealised Scenarios

Channel capacity depends heavily on the scenario, or algorithm, used to manipulate it. In the abstract, it is very difficult (virtually impossible) to show that a particular scenario for manipulating a covert channel

[8] If special diagnostic hardware is available, this may not be an issue.

is optimal. It is much easier to show that every scenario for the exploitation of a certain channel must pay a specific overhead (e.g., deallocating resources that must be allocated to exploit the channel). Some well known overheads, like synchronisation, however, cannot in general be included because it is not known how clever an attacker might be in synchronising the sender and receiver. In general the attacker is assumed to have the use of the entire system (no interference from others). The exploitation scenario can therefore be started by selecting the most efficient TCB primitives for manipulating (sender modifies, receiver observes) the channel, regardless of whether there is an apparent way to use them for that purpose. If the capacity is sufficiently low, the analysis can end there. If the capacity is high, a search can then be performed for reasons why those primitives can't be used, or why other primitives must also be used, lowering channel capacity.

Estimation of Information Rate

Although there is general agreement that information theory (Shannon's definition) is the proper basis for capacity calculations, methods of calculating covert channel capacity is an ongoing research area [21, 13]. Except for some simple classes of channels, precise calculation of covert channel capacity exceeds current mathematical techniques. In order to make calculations feasible, however, simplifying assumptions can be made. By avoiding capacity underestimation, simplifying assumptions sometimes dramatically exaggerate channel capacities.

In the TCSEC, acceptable capacities are expressed for individual channels. In previous evaluations, channel "aggregation" has been an issue. The motivation for aggregating several channels into a single one is the recognition that it may be possible to exploit several channels in parallel, thus increasing the rate at which information is compromised. In the Trusted Xenix evaluation, aggregation was a consideration for channels based on attributes which could be created in large numbers (e.g., directories) by an attacker. For single-processor systems, the effects are essentially to drop context switch time from the capacity calculation. For multiprocessors, aggregation may introduce a factor of n into the capacity estimation where there are n processors (because the channels can be exploited in parallel). Some agreement with the evaluation teams will be required to determine which channels will be subject to aggregation and which will not.

Resource exhaustion (and some policy conflict) channels may be modeled as a one bit noiseless channels. Analytic techniques (and even tools[9]) exist that are adequate to calculate capacities for one bit noiseless channels. An upper bound on the capacities of event count channels can be obtained through simplifying assumptions of the technique used for one bit noiseless channels. Timing channels are more difficult to estimate. Some timing channels operate at memory speeds, limited only by the time required to resolve hardware contention [14]. In this case, the channel is not sustainable using encoding because the sender must "take time out" to encode the information[10], and the analysis can be simplified. Also, contention resolution that is fair in the sense that it does not penalise one symbol or another with a delay reduces the benefits to be gained through coding. Reduced channels will require more careful analysis, however. The following section presents a measurement technique that is useful for many storage channels.

One Bit Noiseless Channels

This section summarises the technique given by Millen [13] for finding the capacity of one bit noiseless channels. A channel may not be noiseless in a real system, but this results only in possible overestimation of channel capacity. Using information theory, the capacity of a noiseless discrete channel is known to be defined by the limit

$$\lim_{t \to \infty} \log_2(N(t))/t$$

where $N(t)$ denotes the number of messages that can be sent in time t. When the effort required to send a 1 is much different from the effort required to send a 0, the capacity significantly exceeds the information rate

[9] Which were used in the Trusted Xenix evaluation.

[10] A consequence of allowing coding in channels that operate at memory speeds is to have channel capacities that *exceed* memory speeds.

obtained when an equal distribution of ones and zeros is assumed. When the transmission of information is effected by a state machine with more than one state, the effort required to send a 1 and to send a 0 may depend on the current state of the state machine. Figure 1 shows a two state machine which corresponds to a one-bit noiseless channel where the edges are labeled by pairs: the symbol before each "/" designates the symbol being transmitted when that edge is traversed, and the letter after the "/" identifies the edge and is a parameter for how much time is required to traverse that edge. The parameters can be understood as follows:

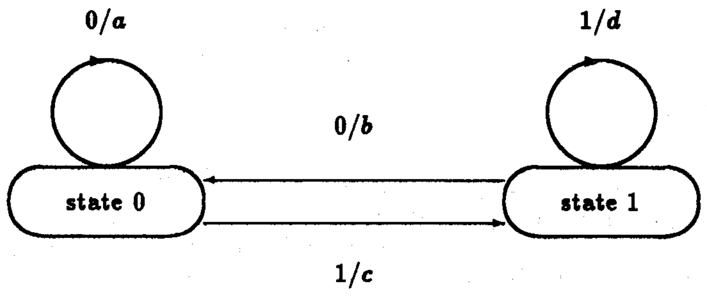


Figure 1: State Diagram For A One-Bit Channel

a send 0 if the last bit sent was 0,

b send 0 if the last bit sent was 1

c send 1 if the last bit sent was 0

d send 1 if the last bit sent was 1

These parameters are related to the definition of channel capacity in [13], where it is shown that the capacity is given by $\log_2(r)$ where $r > 1$ is the (unique) solution of the equation

$$\begin{vmatrix} 1 - r^{-a} & -r^{-b} \\ -r^{-c} & 1 - r^{-d} \end{vmatrix} = 0$$

This equation can be solved numerically given the four state transition times. A more general form, presented in [13], may be applied to state machines which have more than two states and two symbols and can therefore transmit more than one bit at one time. The solution to the resulting equations, however, becomes unworkable when the number of states is much larger than two. In order to measure channel capacity for event count channels, which are modeled as state machines with N states (N possibly large), we can use a simplification which is guaranteed to not underestimate the channel capacity. The simplification finds an upper bound for n-bit channels by always using the smallest state transition time. For N states and N^2 state transitions $s_1, s_2, s_3, ..., s_{N^2}$, $log_2(N)$ bits may be transmitted at one time. An upper bound on the channel capacity is therefore given by:

$$\frac{log_2(N)}{min(s_1, s_2, s_3, ..., s_{N^2})}$$

This upper bound is not tight, but may allow the elimination of some event count channels from further analysis.

Channel Capacity Reduction

If a channel's capacity exceeds acceptable limits, channel capacity must be reduced or audited. Accurate estimation of channel capacity is important because it determines the selection of and severity (performance impact) of reduction techniques. For example, delays that are unnecessarily large degrade system performance unnecessarily whereas delays that are inadequately small affect system security adversely. Some channels may be eliminated by design changes (that usually reduce functionality), or by using certain configuration options. For example, Gemsos allows most storage resources to be statically preallocated by security level, therefore eliminating most resource exhaustion channels. Such preallocation is expensive, however, and primarily addresses a class of storage channels that can be effectively reduced using delays.

Storage Channel Reduction

The two major techniques for reducing storage channels are delay and randomisation. Resource exhaustion channels can be reduced by temporarily suspending (delaying) any process that exhausts a resource. Such delays usually have acceptable performance impact because resource exhaustion is a (relative) rare event for most resources. Delay can be used in a similar way for policy conflict channels. Delay is both less effective and more costly for event count channels that report global status (e.g., total free blocks), however, because: 1) the attribute being observed may take on many values and the receiver therefore may receive more than one bit per delay, and 2) the delay must be imposed on every use of the reporting function.

Event count channels that show how resources are allocated (e.g., new Unix pid's) respond well to randomisation, assuming a sufficiently strong random number generator. For Trusted Xenix, a congruential random number generator seeded by the time of day and number of system calls provided sufficient strength. In practice, an exploiter could not discover the seed because of the frequency and variable number of system calls.

Randomisation is less effective against status reporting event count channels because the accuracy of the functions is inversely related to the degree of "fussing" provided by randomisation.

Timing Channel Reduction

For some timing channels, a system has no way to tell the difference between exploitation and normal activity. This characteristic makes timing channels intrinsically more difficult to reduce than many storage channels. This is particularly true when the channel is based on high speed hardware based contention. The (now classic) example is the shared bus multiprocessor where there are three or more processors [14]. In that channel, low processor A increments a global memory location as rapidly as possible, high processor B sometimes accesses global memory, contending for the bus, and low processor C continually checks the progress of processor A. Bus cycles stolen by B show up (to C) as failures to increment the memory location. This channel operates at memory speeds, and cannot be meaningfully audited by software because the operations used to transmit information are "normal" processing, and because their volume would quickly overwhelm any audit system.

It is beyond the scope of this plan to describe how to delay all timing channels. Several possibilities are:

- Where the system primitives that return the value of a clock can be identified, use delay to reduce the capacity. It is worth noting that the alphabet of such channels may be large, and that the information rate may not be reduced as effectively as it is for resource exhaustion channels (which have an alphabet of {error, not error}).
- Randomly introduce perturbations into readily available clocks to reduce the speed or ease of signaling. Noise may reduce, but cannot close such channels. Analytic techniques for evaluating the effect of the noise may be difficult.
- Fuss some clocks to reduce the accuracy with which covert senders and receivers can measure clock differences. A variant of this approach, used in the VAX security kernel [8], randomised system timers

and added random delays to the initiations and notifications (of completion) of IO. This technique, called "fussy time," attempts to isolate each process from the precise timing information provided by hardware supplied clocks such as interrupts and cpu bus contention. Although the measurement technique was not specified, [8] reports evaluation team agreement that all timing channels in the VAX security kernel were reduced to less than 10 bits per second.

- For contention channels like the bus channel, schedule the resource (in that case, the bus) by security level, so that most contention is limited to being within a security level (and therefore legal). The performance impact of this approach is not known, but may be severe (all processors contending for the bus would have to change security level at the same time).

Assurance of Channel Reduction

Although channel identification may be conducted using specifications, channel reduction techniques must be implemented, and assurance of their effectiveness must be gained at the code level. At the least, some form of covert channel testing must be performed to evaluate the effectiveness of reduction techniques. Code analysis tools may assist for storage channels. Trusted Xenix used "covert channel flow tracing," a method in which function call trees and variable references are analysed to ensure that a delay or randomisation algorithm is always used before selected variables can be reported to a receiving process. IBM did not have a production quality tool and, in practice, performed much of the analysis manually. If the analysis is correctly performed, assurance can be gained in general that storage channels are reduced. It is not clear that such tools can be effective for timing channels, however. Assurance for timing channels may depend on comprehensive testing and code inspection.

Planning the Analysis

The covert channel analysis for a large system should satisfy three goals: 1) proceed concurrently with system development, 2) provide credible results, and, and 3) remain within available resources.

There are basically two approaches to concurrently performing CCA and system development: 1) substantially automate the analysis so that it can be completely redone after each significant system change, or 2) decompose the system into parts each of which can be independently analysed, and then combine the analyses as the system is constructed. In either approach, analysis should be performed continually during development so that feedback from the analysis can impact the system design and implementation.

Although attractive in the abstract, substantial automation of CCA is an area of active research. A number of tools exist that may assist in CCA by automating part of the process or by enforcing rigor in specification: Malpas[12], Ina Flo [5], and an IBM proprietary tool [19] (this list is not exhaustive). In addition, a covert channel analysis tool is under development inside TIS [16]. As is the case with programming projects, the use of such tools may require dramatically more time than is anticipated.

Unfortunately, decomposing the system into components upon which independent CCA can be performed is also a research area. In principle, modular covert channel analysis could be based on Kemmerer's SRM, but there are no worked examples (known to the author). Changes to each component would at the least force reanalysis of the affected component. If the reanalysis changes the results obtained by the previous analysis, other components that depend on the changed component must be reanalysed as well.

Because CCA is still an art, the credibility of the results is somewhat subjective. Clearly an analysis that fails to find many channels that are subsequently discovered in penetration testing or evaluation will not be credible, however. Both specification and code level analyses may miss channels. In general, the rigor imposed by using tools or formal techniques may increase the confidence that specification based analysis is sufficient. It has been claimed that code level analysis finds all storage channels [19][11]. The handling of timing channels will of necessity be informal; here, confidence can be gained only through sustained effort to find as many channels as possible.

[11] However, see section *A Code Level Technique*

Channel Identification

The most fundamental decision for covert channel identification is whether to use noninterference or some form of the SRM methodology (or both). In [7], noninterference was compared with the SRM methodology. Although the authors refrained from selecting one strategy as the best, they noted that noninterference proof failures might become unworkably difficult as the sise of a specification increases. Although, in a high level (and simple) specification, the ideal of noninterference might be reasonable because channels present at that level would of necessity be present in any faithful implementation, a low level specification would (practically speaking) always cause proof failures. The authors further noted in [7] that noninterference, by itself, probably could not be a comprehensive tool, although the SRM might be. Noting that their study was limited, the authors in [7] refrained from selecting one strategy as the "best" and indicated their intention to use both in the future. Unfortunately, a developer must choose a strategy even though there may not be adequate information to show that it is always superior.

Selection of SRM methodology versus noninterference is difficult; in many large scale development activities, however, the following disadvantages of noninterference seem to argue against its use:

- Proofs are difficult; interpreting proof failures is even more difficult.
- Proof failures that are not understood provide no information.
- Noninterference may require a level of formality that cannot be sustained on a large project with many changes to the system.

The following assumes the use of some form of the SRM methodology.

Storage Channels

The primary decision to be made is whether to pursue a code level analysis, an analysis based on specifications, or both. CCA has been more frequently performed on specifications than on implementation code. The considerations can be broken down:

- specification analysis
  - pro
    - easier to do informally
    - potentially less expensive
    - some tools exist (e.g., Ina Jo, Ina Flo[5] [12])
    - the analysis is less sensitive to minor system changes
  - con
    - depends on specification accuracy
    - omits necessary detail—channels not present in the specification will not be discovered
    - there is no way to know when the job is finished (i.e., what specification is low level enough?)
- code level analysis
  - pro
    - includes implementation detail
  - con
    - more expensive
    - few tools, e.g., Malpas [12], are available

[12] Experience on two projects indicates that Ina Flo is not yet mature enough to use.

* tools are required
* because of the complexity of the real implementation, coverage is not likely to be complete—the detail can overwhelm the analysis

The true difference between analysis of specifications and code depends on the amount of detail present in the specifications. Some analyses have used very detailed specifications [11] containing more than 700 state variables. Although there are more "pro" items for the specification approach, the omission of necessary detail and the dependency on specification accuracy are severe handicaps. Equally severe is the great complexity of a code level analysis, in which detail can overwhelm the intuition of the person performing the analysis. Given the limitations and costs of each approach, it is difficult to choose one exclusively. A dual track approach therefore seems most prudent.

A specification analysis should be conducted on the interface specifications, and on each refinement of those specifications. Parallel with that, a code level analysis should focus on validating (not verifying) the correspondence between the specification and the implementation. Although a breakthrough in formal code analysis is possible ([16] may eventually be such a silver bullet), the code level analysis should focus on "informally" validating the specification analysis. Much of the code analysis will probably be manual, but tools to assist the analyst should be obtained or written as necessary. If possible, tools such as Ina Jo and the SRM matrix generator should be used to enforce specification consistency through the provision of type checking, etc., and to construct the SRM.

At the interface level, the first step in the construction of the SRM is to identify the TCB primitives that may be used to manipulate system attributes. Normally this is the TCB interface. It is necessary in the SRM approach for the R and M entries in the cells of the matrix to represent all direct flows between primitives.[13] In this context, two primitives A and B are *atomic* if every interaction between them affects the system state as if they executed sequentially in some order. If two primitives of the SRM were not atomic, then a worst case analysis (including all possible interleavings) would have to be applied to determine what date flows between the two primitives were possible. The kernel calls of some operating systems (e.g., most versions of Unix) provide a simple version of atomicity by suspending most process scheduling during kernel processing. Even with these kernels, some operations will not be atomic because multiple processes may have to be suspended in the kernel waiting for I/O. The analysis should identify what operations are atomic, and how information flows between any non-atomic operations are included in the SRM.

Timing Channels

Identification of timing channels must depend on an informal but extensive search by knowledgeable developers. Wray's methodology can assist in guiding the search for clocks, and the matrix proposed in the methodology can assist the developers in keeping track of the relationships between different identified channels. An approach similar to that used for penetration testing (the flaw hypothesis methodology) may provide the best results. Because timing channels often depend on hardware contention, it will be necessary to conduct the testing on all significantly different hardware platforms (particularly multiprocessors).

Capacity Estimation

CCA for a family of hardware bases should be parameterised by hardware timing characteristics for each supported hardware base. The determination of which channels can be aggregated affects capacity estimation. This determination should be made as channels are identified. Channel capacities for multiprocessor hardwares will require special consideration since the multiprocessor version will probably have more identified timing channels. The timing information can be derived from engineering data or from test programs written to derive the characteristics of each hardware base. The multiprocessor hardware bases will require additional tests to measure characteristics not present in uniprocessors.

As given above, analytic techniques exist for some channel types. For others, upper bounds are required. The use of coding theory is indicated wherever the cost of sending one symbol is much larger than the cost

[13]If a transitive closure is performed, indirect flows would be present as well.

of sending another (perhaps because of a delay). For channels in which all symbols are equally easy to send, the use of coding theory provides little capacity increase, and capacity can be approximated by assuming an equal distribution of output symbols. For such channels, the capacity is $log_2(n) * cycles\ per\ second$ where n is the number of possible output symbols in a cycle.

Storage Channel Reduction

The kind of channel (resource exhaustion, policy conflict, or event count) affects the available alphabet. Exhaustion and policy conflict tend to be binary valued. Event count channels usually have numerous symbols. Some channels can be eliminated through design changes, for example, by removing the status reporting functions, or by changing them to tell white lies. Other channels can be reduced primarily through delay and randomisation. Global identifiers, for example, the process id in Unix, present special problems. They can be reduced using randomisation so long as the space of identifiers is much larger than the number of identifiers that can be in use at any one time, and so long as allocation of the next identifier always chooses randomly from the entire pool of unused identifiers. When caching is used to optimise the use of resources associated with an identifier, the cache reduces the options for selection of the next identifier, and can be exploited to signal. For such global identifiers, the maintenance of separate security level partitions (that move slowly in response to demand) for the identifiers *and* the cache can be used to reduce capacity.

Two attacks on per-process delays must be prevented for delays to be effective: 1) interruption of the delay, and 2) overlapping of multiple delays. If a delay can be interrupted in any way, it is not effective because a process can notice when another process is in a delay, interrupt it, and resume covert communication. It should not be possible to destroy processes that are suspended in delays.

If multiple per-process delays can be overlapped, an attacker may use multiple processes to effectively poll a resource more rapidly than permitted during the delay. This scenario can be prevented by serialising delays. A general serialisation scheme is as follows. Let the delay period be D seconds. The first process to be delayed for use of the channel is delayed D seconds. The second process to use the channel is delayed for the greater of: D seconds or D seconds from the time the first process finishes its delay. Multiple delays for a resource may therefore not overlap. Using this technique, delays can be overlapped when they are imposed on different resources.

Timing Channel Reduction

The suggestions in the above section on timing channel reduction apply as stated. In addition to the use of delays to reduce capacity, however, delays might be used to *hide* activity. For example, to prevent a channel in which one process infers information from another through the time to access a shared page (i.e., whether a page fetch was necessary or not), sporadic delays that would correspond to page fetching could be introduced. The delay must conceal from the receiver the fact that a page fault was not necessary because the sender had already paged in the data. Specifically, all low processing that could not occur during a real page fault must be prevented during a delay that mimics a page fault. If other low tasks could run, the receiver could schedule another task to run and then measure its progress. Processing by higher level tasks could continue, however. Additionally, the delay must be realistic. For example, actually performing a page fault can be expected to take varying amounts of time to account for disk latency, rotational delay, etc. If a delay always takes exactly the same amount of time, but the real operation times would vary, the channel is not effectively reduced.

Recommendations

Covert channel analysis can be approached in the following sequential phases. In each phase, all activities may proceed in parallel:

1. (a) Obtain timing parameters for all hardware bases. Programs that obtain the parameters may be developed on prototype or untrusted versions of the final system.
   (b) Begin the search for timing channels.

(c) Survey available tools for system specification and SRM construction, and evaluate. Select one, or reject all and develop and use a proprietary notation.

2. (a) Continue search for timing channels.
   (b) Complete design documentation to incorporate the use of the selected tool or notation in the specification layers.
   (c) Decide which system components can be independently analysed.
   (d) Begin development of source level tools to support the specification analysis, and also to provide evidence of coverage for channel reduction.

3. (a) Continue search for timing channels.
   (b) Enhance source level tools as necessary.
   (c) Construct the SRM for each system component with a stable interface.

4. (a) Continue search for timing channels.
   (b) Combine analyses of separate components and categorise channels discovered by the SRM.
   (c) Use the source tool to validate the specification analysis.
   (d) Calculate channel capacities (for all platforms, as possible), eliminating from further consideration channels that are too slow.

5. (a) Continue search for timing channels.
   (b) Reduce or audit identified channels through system source or configuration changes.
   (c) Use the source tools to check coverage of reduction techniques.

6. (a) Continue search for timing channels.
   (b) For all changed components, until the system is frozen:
      i. Recalculate the SRM (or determine informally that it need not be recalculated).
      ii. Revalidate the SRM using source tools (incrementally, if possible).
      iii. If any new channels are discovered, calculate their capacity, and reduce or audit as necessary and possible.

The search for timing channels is present in each phase, but the effort required in each phase may not be equal. The search for timing channels should be performed until the number (and severity) of additional channels discovered using a given amount of energy falls below some threshold. Because system changes can introduce new channels, the search must be revisited until the system is frozen (but perhaps at much reduced levels of effort).

The CCA will require a diverse set of skills: 1) skills in the use and evaluation of tools (including an understanding of formalism), 2) coding skills, 3) knowledge of the role that covert channels played in past evaluations, and 4) design knowledge of the system being analysed. The writing of test programs and the search for timing channels can contribute to design knowledge. The covert channel "team" should include trust engineers and developers.

It is important to allocate sufficient energy for these tasks. The energy devoted to CCA will be used to evaluate tools, create (modest) tools, write test programs, perform analysis on a complicated body of changing software, produce designs to reduce and audit identified channels, and achieve assurance that identified channels are reduced. *This is an enormous amount of work and should not be underestimated.*

Acknowledgments

The author would like to thank Mike Masurek, Tim Redmond, and the reviewers for helpful comments on the technical content and presentation.

*

References

[1] National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

[2] L. Badger, F. L. Mayer, T. Redmond, "Trusted Xenix Covert Channel Capacity Estimation And Reduction," TIS Technical Report #364.

[3] D.E. Bell and L. Lapadula, *Secure Computer System: Unified Exposition and Multics Interpretation.* (Technical Report No. ESD-TR-75-306, Electronics Systems Division, AFSC, Hanscom AF Base, Bedford MA, 1976).

[4] D. E. Denning and P. J. Denning, "Certification of programs for secure information flow," Communications of the ACM, (July, 1977), pp. 504–513.

[5] S. T. Eckmann, "Ina Flo: The FDM Flow Tool," Proceedings of the 10th National Computer Security Conference, p. 175-182.

[6] J.A. Goguen and J. Meseguer, "Unwinding and Inference Control." Proceedings of the 1984 IEEE Symposium on Security and Privacy, 1984.

[7] J. T. Haigh, R. A. Kemmerer, J. McHugh, and W. D. Young, "An Experience Using Two Covert Channel Analysis Techniques on a Real System Design," IEEE TSE Vol. SE-13, No. 2, Feb. 1987.

[8] W. Hu, "Reducing Timing Channels with Fuzzy Time," Proceedings of the 1991 Symposium on Research in Security and Privacy, May, Oakland, Cal.

[9] Information Technology Security Evaluation Criteria, Harmonised Criteria of France, Germany, the Netherlands, and the United Kingdom May 1990.

[10] R.A. Kemmerer, "Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels," ACM Tran. on Computer Systems, 1:3, August 1983, p 256-277.

[11] T. E. Levin, A. Tao, S. J. Padilla, "Covert Storage Channel Analysis: A Worked Example," Proceedings 13th National Computer Security Conference, Oct. 1990.

[12] Malpas: a commercial tool used on the VAX security kernel that can assist in code level analysis for multiple languages: Pascal, PL/1, Fortran, and C.

[13] J.K. Millen, "Finite-State Noiseless Covert Channels," Proceedings of the Computer Security Foundations Workshop II, Franconia, New Hampshire, P.81, 1989.

[14] "Minutes of the First Workshop on Covert Channel Analysis," September 19-21, 1989.

[15] T. Redmond, B. A. Mayer, F. L. Mayer, "The Abstract TMach Kernel Model," TIS Technical Report #293.

[16] T. Redmond, "Formal Approach to Identifying Covert Channels in Source Code", TIS Technical Report (in progress).

[17] C. E. Shannon and W. Weaver, "The Mathematical Theory of Communication," the University of Illinois Press, Urbana, Illinois, 1964.

[18] D. Sutherland, "A Model of Information," Proceedings of the 9th National Computer Security Conference, Sept. 1986, p. 175.

[19] C. Tsai, "A Formal Method for the Identification of Covert Storage Channels in Source Code," Proceedings of the 1987 symposium on Research in Security and Privacy, May, Oakland, Cal.

[20] C.R. Tsai, "Covert-Channel Analysis in Secure Computer Systems," Phd. Dissertation, University of Maryland, College Park, Maryland, Aug. 1987.

[21] J. T. Wittbold, "Information Flow in Nondeterministic Systems," Proceedings of the 1990 Symposium on Research in Security and Privacy, May, Oakland, Cal.

[22] J. C. Wray, "A Methodology for the Detection of Timing Channels," Proceedings of the Computer Security Foundations Workshop II, Franconia, New Hampshire.

[23] J. C. Wray "An Analysis of Covert Timing Channels, " Proceedings of the 1991 symposium on Research in Security and Privacy, May, Oakland, Cal.

DEALING WITH MALICIOUS LOGIC THREAT
A PROPOSED AIR FORCE APPROACH

Howard L. Johnson
Information Intelligence Sciences, Inc.
1903 So. Franklin St.
Denver, CO 80210
(303) 777-4266

Chuck Arvin and Earl Jenkinson
CTA Incorporated
7150 Campus Drive, Suite 100
Colorado Springs, CO 80918
(719) 590-8890

Captain Bob Pierce
AF Cryptologic Support Center
Hq. Electronic Security Command, AFCSC/SR
Kelly AFB, TX 78243-5000
(512) 925-2511

ABSTRACT

Trojan horses, viruses, worms, and other malicious logic that seek to interrupt service or modify or destroy data are not necessarily defeated by confidentiality mechanisms. The Air Force Trusted Critical System Evaluation Criteria (AFTCCSEC) [1] supplements confidentiality requirements found in the Trusted Computer System Evaluation Criteria (TCSEC) [2] by addressing integrity and service assurance. This paper introduces and describes criticality division/class G2 found in the AFTCCSEC. The approach imposes mandatory controls including access constraints, type enforcement, detection techniques, and use of a resource scheduling architecture. It applies to all life-cycle phases: development, distribution, operations, and maintenance. Features include program/data isolation (e.g., physical, logical or use of cryptography), protection against covert criticality channels (that allow malicious code insertion), and strict configuration control of software and hardware. Any TCSEC division/class and G2 criticality can coexist, though retrofit of G2 will require an existing TCSEC TCB of B1 or higher. This paper provides a basic understanding of the concepts and policy, and also addresses questions most often asked by reviewers of the AFTCCSEC document.

THE PROBLEM

Compromise of classified information has been the primary concern of DoD computer security for three decades. The viruses and Internet worms have shown the reality of malicious code attacks.

Work was accomplished under CTA Contract Number F41621-88-D5001 issued by Hq. Electronic Security Command, AFCSC/SR, Kelly AFB, TX 78243-5000.

There have been no previous DoD requirements, evaluation criteria, and models that specifically address the malicious logic problem.

Thus, systems that previously were deemed secure according to TCSEC requirements may in fact be vulnerable. Because upgrade message flow is usually allowed, this could theoretically provide the ability to insert and execute malicious programs. A likely attack involves inserting a small virus to monitor a compromised channel. It would recognize other malicious code hidden between communications protocol "end" codes in incoming messages and cause its execution. It might then erase any trace of the larger code. The enemy would have subversion capability over the life of the system (or until thwarted) to perform subversive missions (fact finding, sabotage, or attempting to leak classified data). The attackers would make these activities difficult to detect or to distinguish from other failures types.

There is a sense of urgency to provide defenses against potential debilitating malicious logic attacks on major command and control systems. We believe the best immediate defense is to provide quick, substantial reaction to the threat.

THE NATURE OF ERRORS AND FAILURES

When a computer system error failure is discovered, it is often not immediately known whether the cause is hardware, a design/ development error, an accident, or a malicious attack. An error or an accident may result in a normal or simple failure, a failure that propagates, or one that exhibits nonpredictable (chaotic) behavior [3]. The most common state is "normal" (see for example Beizer 1983 [4] for references) although people like to talk about the exceptions (the 1989 AT&T failure Neumann [3]). An accident has no goal so one would expect the impact to a system to be naturally (e.g., normally) distributed. A malicious intruder (not a harmless hacker) will often seek debilitating system impact.

Malicious logic is generally more complex than an accident or an error. Accidents and errors are seldom caused by more than a single action and or flaw. An accidental action or flaw can normally be emulated by a few computer instructions. The length of known virus and worm attacks, however, is generally on the order of 1000 or more bytes. Sixty percent of reported viruses are derived from the Jerusalem B virus which is approximately 1800 bytes. Others range from 405 bytes (405 Virus) to 60,000 bytes (the Internet Worm). The reason code for a malicious attack is large is because the perpetrator usually has multiple objectives that include detection avoidance, formatting to conform to applications, causing a state of quiescence, planning, file searching, communications monitoring, trigger monitoring, self erasure, and/or propagation.

Design/development errors exist prior to and after validation (if not discovered) and generally repeat (e.g., after rollback). Hardware failures occur after validation and may be transient. If they repeat, they can be caught by diagnostics. Accidents seldom repeat and are usually recognized by individuals involved after they occur. Malicious attacks can occur either prior to or after validation, though avoiding a thorough validation is difficult. Malicious attacks can repeat, may not repeat, are probably not revealed by diagnostics (but could be if the attacker desired), often have multiple stages, and sometimes give multiple independent results. Joseph and Avizienis [5] propose a logic tree approach that assists in determining the cause of an error or a failure.

SCOPE

In recent papers (e.g., [6]) we defined a need and an approach to deal with loss of integrity and denial of resource use. This evolved into the Air Force Trusted Critical Computer System Evaluation Criteria (AFTCCSEC) patterned after the Orange Book (TCSEC). The AFTCCSEC has been published as Air Force Special Security Instruction (AFSSI) 5029. Figure 1 shows the division/classes of the AFTCCSEC and the relationship to the TCSEC D and C1 levels. This paper focuses on criticality class G2 that incorporates protection against malicious logic. Class G3 which addresses critical Air Force systems and classes F3, F2, F1 and E1 that address multilevel systems and higher assurance, are discussed in other papers [7 and 8]. The AFTCCSEC basically uses the TCSEC control objectives. A reinterpretation is required since AFTCCSEC addresses integrity and service assurance which complements the TCSEC application to confidentiality.

Criticality Division/Class	Protection
H	Same as sensitivity D
G	Single Level
G1	Almost the same as sensitivity C1
G2	Protects against malicious logic
G3	Supports Critical operations
F	Multilevel (Labels)
F1	Critical and Highly Critical
F2	Critical and Non Critical
F3	No clearance and Critical
E(E1)	Formal methods (no clearance and Highly Critical)

Figure 1 AFTCCSEC Division/Class

APPROACH

Current DoD budgets cannot afford to duplicate present Orange Book security costs. Therefore, in the AFTCCSEC we have taken an approach that has three implementation cost reducers. Each also reduces time until implementation and implementation risk.

a) Division/class required depends on mission criticality. Malicious logic protection is introduced at the G2 level where systems are neither critical nor highly critical and can be realized with a minimum fund expenditure.

b) Since the approach follows directly from the TCSEC, most mechanisms and procedures required by the TCSEC can be used directly or modified to accommodate AFTCCSEC requirements.

c) Cryptography and cryptographic checksums used as isolation mechanisms will reduce vulnerability and cost of protection.

POLICY

This section reviews new policy proposed for the Air Force:

There shall be protection against malicious logic throughout the system life-cycle beginning with development and continuing through assurance, distribution, operations, and maintenance.

COMPUSEC techniques used by the TCSEC for discretionary access control, object reuse, accountability, assurance, and documentation shall be used where possible for program and data integrity and assurance of service protection.

COMPUSEC techniques shall be employed using Air Force accepted trusted approaches to control access by individuals and processes to programs (stored processes), data, and system resources.

Intrusion detection shall be used to discover unauthorized users, system misuse, or malicious logic. Response should include fault isolation, analysis, and malicious logic elimination. These capabilities shall be protected from malicious logic attacks.

Public and private key encryption, and cryptographic checksums shall be used for the protection of data and programs where technically feasible and when cost, performance, and risk requirements can be met. (Standards shall be developed that relate the strength of algorithms and key management approaches to the protection required, supplementing current use of encryption to protect classified and sensitive information.)

Information gained from traffic analysis shall not reveal knowledge of system or security protection details that could be used in a malicious attack.

Software shall be developed, stored, and delivered under strict configuration control and screening to make the probability of malicious hardware or software reasonably small.

PREVENTION APPROACH

As stated in policy, the AFTCCSEC uses techniques identical to the TCSEC including the trusted computing base (TCB), discretionary access control, object reuse, accountability, assurance, and documentation. Additional or changed techniques introduced at the G2 level are discussed further.

Constrained Access

Current security mechanisms control access of subjects to objects. Constrained access (Figure 2) adds one dimension to the access control process by constraining process access to objects, independent of user. Specific access type is also controlled (called type enforcement). A process must be on a valid process list to be executed and can be removed from that list to quickly contain malicious code. Constraints are identified by way of process and object profiles. Processes are restricted to interact as they were intended when programmed. Additional constraints restrict operations on objects to the minimum required subset. Constraints are identified by the developer and established by the security officer. Attempts to violate the restrictions are reported to the auditing function. There is strict configuration control of programs, constant data, valid process lists, process profiles, and object profiles throughout the system life-cycle to detect unauthorized modification or other potential malicious characteristics. The idea of a security policy between users, processes, and objects, (also called triplets) is discussed by Clark and Wilson [9] and the control by access type (also called type enforcement) is discussed by Boebert and Kain [10].

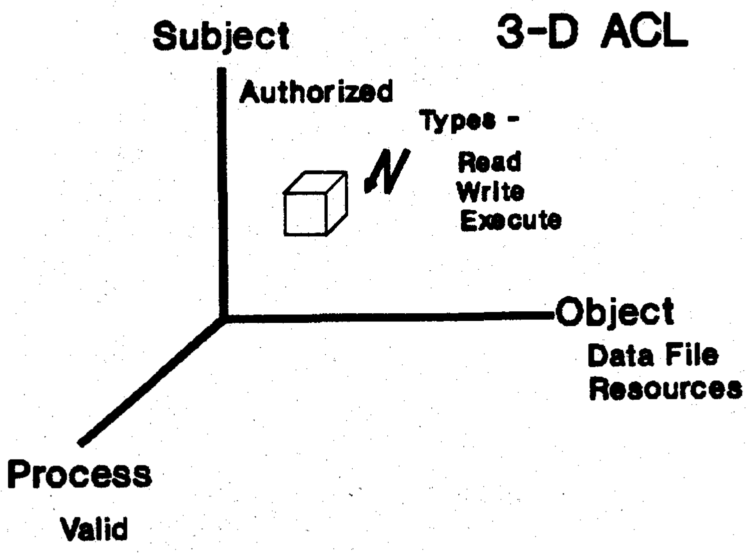


Figure 2 Access Control Triplets

Covert Channels

A covert channel is a communication channel that allows unauthorized transfer of data in a manner that violates security policy. A sensitivity attack using a covert channel has three steps and a criticality attack has two steps as shown in Figure 3. In the TCSEC, the protection objective is to control data leakage. The TCSEC does

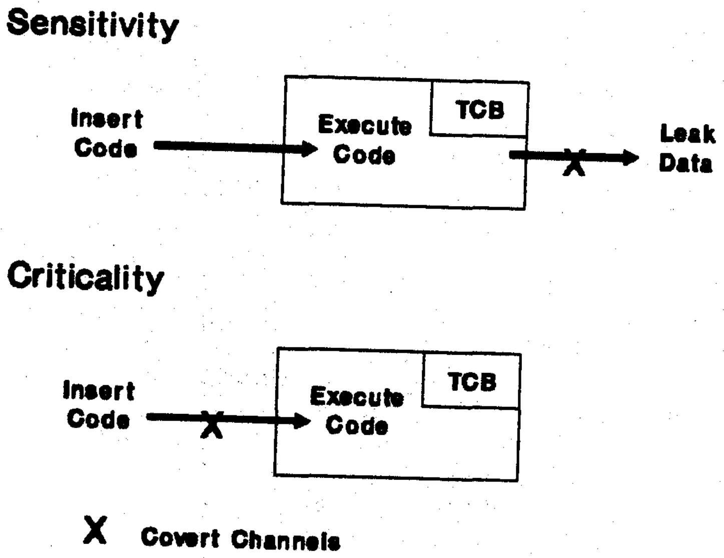


Figure 3 Covert Channels

little, except through discretionary security, to control insertion and running of malicious code.

In criticality, covert channels are considered at the G2 division/class. Input data must be assured to be malicious logic free. Unauthorized channels "in" are potential covert channels that must be plugged or monitored and are of concern during development, delivery, operations and maintenance. AFTCCSEC covert channel methods are much the same as the TCSEC.

Cryptography

Cryptographic processes protect data from vulnerabilities in trusted domains or when data is traveling through untrusted domains. Private keys and private encryption algorithms are controlled by the TCB. Private key encryption prevents unauthorized reads and executes and some algorithms detect data modification. Decryption can invalidate unencrypted malicious logic (see the Pozzo-Gray Virus Containment Model [11]). Cryptographic checksums detect unauthorized modification. Public key encryption identifies the originator and, when used with a checksum, allows users even in an untrusted domain to detect modification. One-way encryption can be used for identification/ authentication. Useful cryptographic processes are itemized in Figure 4.

Non Disclosure	**Bandwidth Filling (Covert Channel)**
Identification	**Execution Prevention**
Key Management	**No Intelligent Change**
Labeling	**Enemy Spoofing**
Mechanism Protection	**Signature**
Modification Detection	

Figure 4 Encryption Uses

AFTCCSEC requirements can be implemented with or without cryptographic processes. The intent is to open the door to cryptography use for other than confidentiality. Cryptography is efficient and inexpensive, and will become even more so as popularity is gained. The issue of required strength can be raised during design and dealt with by the appropriate DoD organizations.

Criticality TCB

The TCB for integrity and denial of service protection is larger and more complex than required by the TCSEC. Some of the functions (e.g., encryption) will normally be implemented in hardware. The primary increases in complexity are for detection and resource scheduling. Protocols, constant data, programs, and other control data are protected by the TCB by ensuring against unauthorized modification using cryptographic checksums. Cryptographic processes are essentially an extension of the TCB.

Trusted Distribution

The TCSEC is concerned about someone tampering with the TCB. The AFTCCSEC additionally worries about injection of malicious logic to system hardware, firmware, or software. Downloading of software within a complex system is also considered a distribution problem.

DETECTION APPROACH

Different than confidentiality, in preserving integrity and assuring service, an effective approach is to detect a problem and respond in adequate time (called critical time) to ensure the mission is still accomplished. At the G2 level, missions are not critical, however, detection is still based on a response time model. Assuming the malicious logic has avoided or defeated prevention mechanisms, the strategy is to identify the occurrence of a malicious attack, minimize its impact, and make the required correction (e.g., remove malicious logic).

Real-Time Audit

Malicious attack detection uses both an inductive and a deductive approach. The inductive approach determines intrusion behavioral characteristics and seeks them out. The deductive approach determines the normal behavior of many aspects of the system through statistics and use of profiles to help determine what is abnormal behavior. In each case a discrimination technique is used to reduce false alarms. This approach makes use of current intrusion detection research (presented by Lunt [12]) in application of statistical, rule based, expert, and other heuristic approaches. Nothing previously unproven is required by the AFTCCSEC, and the door has been left open to technological advances.

To avoid overhead, auditing can be accomplished in parallel by low-cost, high-performance hardware. Auditing may be thought of as a time prioritized data driven process. An audit function is triggered by the availability of its applicable detection/audit data. The maximum time until execution is determined based on the time variables specified by the policy. The function and time are placed in a time prioritized queue. The time is counted down and the function with minimum time is executed. The detection process checks itself for a possible denial of service attack and responds with a corresponding predefined response plan. Data compressing and discarding can be used.

Resource Scheduling

A precise resource scheduling policy must be defined, both to define what constitutes denial of service and to know what action must be taken in response to a denial of service attack.

Malicious Code Search

A tool that searches for malicious logic can be used during development as part of validation and during operations as part of configuration control, real-time audit, and communications monitoring. Search profiles help to recognize known or modified malicious logic, illegal system functions, or system-only functions. Non random data in encrypted (random) data streams can also be identified. Keeping the search profiles secret and the search process protected increases the mechanism effectiveness.

Hardware pattern matching logic can perform a fuzzy search. The term "fuzzy" means that the profiles need not match exactly. Application specific frequency weighting can be used to further discriminate. Hardware implementation can reduce search of very large databases to a few hours and keep up with very high communication bandwidths.

SUMMARY

Current approaches in PCs to virus prevention, detection, and isolation/removal are an ever growing compilation of checks that a clever infiltrator eventually can work his way around. The philosophy of playing catch-up will always leave the penetrator with the advantage. That approach presumes repetition or variations on past attacks. The professional infiltrator will probably not use known malicious code.

The approach in the AFTCCSEC contains as a minimum all of the known protections used by antivirus software. The approach further depends on the existence of a TCB and utilizes strong encryption. The approach allows the protection to be site, application, and security officer specific, avoiding the predictability of canned solutions.

This paper has presented the policy and discussed new approaches introduced at the G2 division/class of the AFTCCSEC to deal with the malicious logic problem for DoD systems. The approach has used the concepts, mechanisms and language of the Orange Book (TCSEC) to simplify understanding and reduce implementation cost. The approach can be implemented in an Orange Book protected system or one where confidentiality is not an issue.

GLOSSARY

Constrained Access Control- A security policy that identifies which processes may be executed and what objects (i.e., other processes, storage objects, and I/O devices) they may access. Process and object profile data are used to ensure that each process access of an object is allowed and is of the allowed type.

Denial of Service - Action or actions that result in the inability of the system or any essential part to perform its designated mission either by loss or degradation of operational capability.

Integrity - Ensuring that data changes in only highly structured and controlled ways. Air Force regulations define integrity as a computer security characteristic that ensures computer resources operate correctly and that data in the data bases are correct. The integrity protection goal is to protect against deliberate or inadvertent unauthorized modification or execution.

Malicious Attack - Insertion of malicious logic, exploitation of system flaw (e.g., trapdoor), or protection mechanism bypass. The attack is considered a fault which may or may not result in an error.

Malicious Logic - Computer hardware, firmware, or software intended to do harm to the system, its data, or the mission being supported.

Object Profile Data - An access control list of processes (programs), the objects for which they are authorized access, and their access type.

Process - A program that has been requested to be executed. It is completely characterized by a single current execution point (represented by the machine state) and address space. The process becomes an entity once it is recognized by the Trusted Computing Base (TCB) that it is potentially to be run (e.g., executed). A process that is not part of the TCB is an internal subject.

Process Profile Data - Identifies legitimate objects (files, resources, and programs) to be accessed by processes and access type.

Program - An object containing potentially executable computer instructions.

Service Assurance - Ensuring availability of a system disrupted by malicious or nonmalicious errors or failures where availability is defined as the computer security characteristic that makes certain computer resources are available to authorized users when needed.

REFERENCES

[1] AFSSI 5029, Air Force Trusted Critical Computer System Evaluation Criteria, Air Force Special Security Instruction 5029, Air Force Cryptologic Support Center, June 1, 1991

[2] DoD 5200.28-STD, Trusted Computer System Evaluation Criteria, December, 1985

[3] Neumann, P.G., "Toward Standards and Criteria for Critical Computer Systems," Compass '90, Proceedings of the Fifth Annual Conference on Computer Assurance, 25-28 June 1990, NIST and IEEE, pp. 186-188

[4] Beizer, Software Testing Techniques, Van Nostrand, 1983, p. 35

[5] Joseph, M.K., and A. Avizienis, "A Fault Tolerant Approach to Computer Viruses," Proceedings 1988 IEEE Symposium on Security and Privacy, 18-21 April 1988, pp. 52-58

[6] Johnson, H.L, "Security Protection Based on Mission Criticality, Proceedings Fourth Aerospace Computer Security Applications Conference, IEEE, December 12-16, 1988, pp.228-232

[7] Johnson, H.L., C. Arvin, E. Jenkinson, B. Pierce, "Proposed Security for Critical Air Force Missions," Information Intelligence Sciences, Inc, February 15, 1991

[8] Johnson, H.L., C. Arvin, E. Jenkinson, B. Pierce, "Proposed USAF Approach to Multilevel Criticality," Information Intelligence Sciences, Inc, February 15, 1991

[9] Clark, D.D, and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," Proceedings of the 1987 Symposium on Security and Privacy, Oakland, CA., April 1987, pp. 184-194

[10] Boebert, E. and R. Kain, "A Practical Alternative to Hierarchical Integrity Policies," Proceedings 8th National Computer Security Conference, 30 September 1985

[11] Pozzo, M.M., and T.E. Gray, "Computer Virus Containment in Untrusted Computing Environments," Information Security: The Challenge, preprints of papers from the Fourth IFIP Security of Information Systems Conference, Monte Carlo, December, 1986

[12] Lunt, Theresa "Survey of Intrusion Detection Approaches," Proceedings 11th National Computer Security Conference, NBS and NCSC, 17-20 October, 1988

[13] AFR 205-16, Security: Automatic Data Processing (ADP) Security Policy, Procedures and Responsibilities, Department of the Air Force, April 28,1989

Developing Applications on LOCK*

Richard O'Brien and Clyde Rogers
SCTC
1210 W. County Road E., Suite 100
Arden Hills, MN 55112

Abstract

The Logical Coprocessing Kernel (LOCK) system is a highly assured INFOSEC system that can be used as a platform to develop countermeasures to current and future security threats. In this paper we discuss the manner in which applications are developed on LOCK and the features of the LOCK system that allow these applications to be developed quickly and securely. The paper focuses on the design of such applications using LOCK's type enforcement and the implementation of these applications using the current LOCK software development environment.

1 INTRODUCTION

The Logical Coprocessing Kernel (LOCK) system is a highly assured INFOSEC system that can be used as a platform to develop countermeasures to current and future security threats. The system is based on a trusted computing base (TCB) that satisfies the security requirements defined for the A1 level in the *Trusted Computer System Evaluation Criteria* [1] and includes embedded encryption for media storage. The LOCK design uses a security coprocessor, called the SIDEARM, that makes access decisions based on conventional multilevel and discretionary security mechanisms as well as LOCK's unique type enforcement mechanism. The SIDEARM attaches to a host processor and, together, the two processors define and enforce the system's security decisions [2], [3], [4].

The approach taken in the design of the LOCK system is based on the belief that the threats that a computer system faces are constantly growing. As more secure computer systems are developed, techniques for attacking these systems are also being developed and becoming more sophisticated. In order to counter these new threats, LOCK is based on an open security architecture that allows for the development of additional security countermeasures as the need arises. In this paper we discuss the manner in which applications are developed on LOCK and the features of the LOCK system that allow these applications to be developed quickly and securely. The paper focuses on the design of such applications using LOCK's type enforcement and the implementation of these applications using the current LOCK software development environment. We also describe future enhancements to the software development environment.

In section 2, a brief description of type enforcement is presented, and section 3 then describes some ways in which applications can be designed to take advantage of the enhanced security and integrity provided by type enforcement. A description of LOCKix, LOCK's version of Unix[1], and the manner in which applications can currently be implemented on LOCK using either LOCKix or the LOCK TCB interface is presented in section 4. Future enhancements that will provide additional support for implementing privileged applications are described in section 5, and section 6 gives examples to illustrate these ideas.

*©1991 SCTC. All Rights Reserved.
[1]Unix is a registered trademark of AT&T

2 Type Enforcement

LOCK provides a *type enforcement* mechanism, used to restrict the access of subjects (processes) to objects (data) and other subjects. In contrast to discretionary access mechanisms, which can be circumvented, type enforcement supports mandatory controls which provide assurance equivalent to that provided by the multilevel controls. Type enforcement controls are orthogonal to multilevel controls, and provide separation and security both within and across levels. In this section, we present a brief review of the type enforcement concept. More details can be found in [5]. A comparison of the type enforcement mechanism with the ring mechanism of Multics can be found in [6].

The LOCK type enforcement mechanism associates a *type* with each object and a *domain* with each subject on the system. The access a subject is permitted to an object depends on the access capability that the subject's domain is permitted to the object's type. Further, the access a subject is permitted to another subject depends on the access capability that the first subject's domain is permitted to the second subject's domain.

Conceptually, the access a subject has to an object via type enforcement can be thought of as an entry in a data structure called the Domain Definition Table (DDT). The DDT is a matrix with columns indexed by type and rows indexed by domain. Figure 1 shows a portion of a sample DDT and lists the possible capabilities a subject can be granted to an object. The matrix entry in the (d,t) position contains the access capability a subject in domain d is permitted to an object of type t. Similarly, the access capability that one subject has to another subject via type enforcement can be thought of as an entry in a data structure called the Domain Interaction Table (DIT). The DIT is a matrix with columns and rows both indexed by domain. The matrix entry in the ($d1$, $d2$) position contains the access that a subject in domain $d1$ is permitted to a subject in domain $d2$. The subject to subject capabilities are: observe, signal, create, and destroy. Trusted capabilities are defined for each access capability that involves modification: trusted write, trusted create, trusted destroy and trusted signal.

The LOCK type enforcement mechanism can be used to solve security problems not addressed by the multilevel and discretionary security policies. It can also be used to develop high integrity subsystems. The manner in which this is done is described more completely in section 3.

3 Designing Applications that Use Type Enforcement

Designing a LOCK application adds a major step to a developer's software design process. Rather than just decomposing the application along functional lines, it must also be partitioned along security and integrity lines. The application designer must identify the components of the application that require added security or integrity, and modularize the application to isolate those components in separate subjects. The collection of subjects that make up an application are called a software *subsystem*.

The design goal is to put each different security or integrity relevant task into its own subject that runs in a distinct domain, and to isolate the data that these subjects must handle into special types. Only the appropriate access capabilities that a subject in each domain requires to perform its task and to communicate with other subjects are assigned to the domain via the DDT and DIT. Then, rather than calling a function to perform a security relevant task, a subject sends a message to an isolated subject designed to perform that task and waits for a return message.

A number of design concerns may require parts of a system to be modularized and isolated. In this section, we discuss some of these concerns and describe how type enforcement can be used to address them.

Subsystem Separation

As part of a subsystem design, special types for subsystem objects and special domains for subsystem subjects are generally defined. The degree and manner of interaction between the subsystem and other subsystems can be rigidly controlled by the DDT and DIT configuration. If

Type / Domain	. . .	UnF1 Data	F1 Data	F1 Code	TrP1 Data	TrP1 Code	. . .	DB Data	DB Code
Pre F1		r, w c, d	-	-	-	-		-	-
F1	-	r	r, w c, d	e	-	-	-	-	-
TrP1	-	-	r	-	r w, tw c, tc	e	-	-	-
⋮			-	-	r, d	-		-	-
DB	-	-	-	-	-	-	-	r, w c, d	e
⋮			-	-	r, d	-		-	-

Figure 1: A Sample DDT. Domains are listed down the left-hand side, and types are listed across the top. The capabilities are: r - read, w - write, a - append, e - execute, c - create, d - destroy. Trusted capabilities grant the domain the privilege of violating the *-property in a well-defined fashion for objects of the given type. The trusted capabilities are: tc - trusted create, tw - trusted write, td - trusted destroy. A dash, -, indicates that the domain is not allowed any access to the type.

total isolation of the subsystem files from other system subjects is desired, then the DDT can be configured so that subjects that are not in one of the subsystem domains are not allowed access to objects of the subsystem types. Hence, no subject outside of the subsystem can access the subsystem's data. Similarly, subjects within the subsystem can be prevented from accessing data outside of the subsystem. The DB domain and its corresponding types, DB data and DB code, in Figure 1 is an example of a subsystem that has been completely isolated from the rest of the system by the proper configuration of the DDT.

The DDT and DIT can also be configured so that communication between different subsystems can only occur through a well-defined interface. For example, a subsystem can have a message queue of a special type that provides the only means for subjects outside the system to contact it. Access to this message queue can then be limited to subjects in special domains.

Managing Trust

Trust on the LOCK system has a very specific meaning. It can be used to override the *-property and permit a subject to modify (write, append, create, destroy) a lower level object, or modify (signal, create, destroy) a lower level subject. It is implemented and enforced using the type enforcement mechanism by defining special domains that have *trusted* access capabilities to objects of special types. A subject in one of these domains has the privilege to perform trusted accesses.

Note that only accesses that involve modification have trusted modes. Accesses that involve observing (such as read and execute) have no trusted mode on LOCK. There is no privilege that allows a lower level subject to read higher level data.

LOCK's approach to trust provides a number of design and security advantages. Trust can be granted at a very fine granularity in conformance with least privilege. Since there are separate trusted accesses for each mode of modification, only the access that is required needs to be granted. Furthermore, the DDT can be configured so that these accesses are only granted to special domains and types. That is, for objects trust is granted on a domain-to-type basis, and granting a trusted access to objects of a given type does not mean that such access is also granted to objects of other types. Similarly, for subjects, accesses are granted on a domain-to-domain basis. Hence, even if a subject has a trusted access, it can only use this access on objects of the indicated type, or subjects of the indicated domain. Since those subjects that use trust are specifically identified and isolated, a least privilege policy with respect to the use of trust can be implemented.

(In this paper we use the term *privileged* subject to indicate a subject that is intended to perform some security or integrity critical function. This is what often is called a *trusted* subject. We use the term privileged, rather than trusted, to avoid confusion with the more restricted notion of trust, described above, that involves the ability to override the *-property of the Bell and LaPadula model. We will restrict our use of the phrase trusted subject to indicate a subject whose domain has a trusted access capability.)

Separation of Duties

Within a subsystem, the LOCK type enforcement mechanism allows a strict *least privilege* policy to be implemented and enforced. In order to take advantage of this capability, the subsystem must be designed in a modular fashion that isolates privileged functionality in separate modules. These modules can then be implemented as separate subjects, each in its own special domain, and the data that they access can be assigned special types. The assured pipelines, described in the next section, are examples of such design. The DDT and DIT can be configured to allow only the least amount of access necessary for the desired functionality. In particular, individual subsystem modules can be prevented from accessing data or communicating with other subsystem subjects in ways that are unnecessary for the proper function of the module.

Such a design allows for simple modifications and additions. Adding a new subject to perform a new task is a localized operation, so its effects on system security and integrity can be easily identified. Also, such a design simplifies assurance work by identifying and isolating security and integrity critical subsystem portions. The primary assurance effort can then be directed toward only those subjects that perform privileged tasks.

Unbypassable Filters

The type enforcement mechanism also provides a means for implementing high integrity operations. By using special domains and types, filter processes can be created to strictly control the manner and order in which certain operations are performed. As figure 2 indicates, these filters have the three critical properties of a reference monitor. They are unbypassable, tamperproof, and can be verified correct. These properties are implemented by the definition of the necessary types and domains and by the correct configuration of the DDT. In Figure 1, the F1 and the TrP1 processes are examples of unbypassable filters.

By composing one or more such filters, *assured pipelines* can be constructed that ensure the security and enhance the integrity of data that flows through the pipeline. This is illustrated in figure 3. Assured pipelines and the LOCK concept of a role, described below, can be used to implement a variety of integrity policies, including those proposed by Clark and Wilson [7], [8]. One application of an assured pipeline might be to guarantee that any modifications to user records must pass through a previewer pipeline before the modifications are committed. This previewer pipeline allows the user to review and commit the changes using filter processes that have been assured to maintain certain integrity properties of the records.

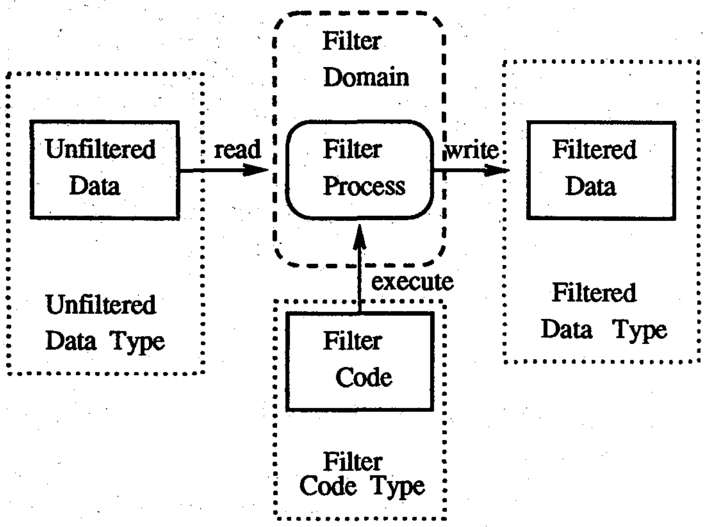


Figure 2: A filter process. The filter reads the unfiltered data and performs its filtering operation before writing the data to a new object of a different type. A special domain (the Filter Domain) is created for the filter process and special types are created for the unfiltered data (Unfiltered Data Type), the filtered data (Filtered Data Type), and the filter code (Filtered Code Type). By using the DDT to restrict create and write access to Filtered Data Type objects to the Filter Domain, the filter process is made unbypassable–it is only through the Filter Domain that filtered data can be produced. By allowing the Filter Domain execute access to only objects of Filter Code Type and by not allowing any other domain create or write access to objects of Filter Code Type, the filter process is made tamperproof. (There is no way to modify the code that it executes.) By having only one object of type Filter Code Type and performing the desired assurance on that code object, the filter process can be verified to perform its filtering process correctly.

Roles

In the LOCK system, user roles are implemented in a manner that relies on the use of types and domains. Every subject is associated with a user. A Role Authorization Table is used to determine in which domains each user is allowed to have subjects operating. Roles are represented as sets of domains, and a user is allowed to operate in a particular role (or subrole) only if the Role Authorization Table permits the user to have subjects in the domains associated with that role (or some subset of these domains).

To extend the example from the previous subsection, the Role Authorization Table can be configured so that only users identified as System Security Officers (SSOs) are allowed the ability to have subjects in the previewer filter domain. In this way, only an SSO is allowed to modify LOCK user records.

4 Implementing LOCK Applications

After developing a design that takes advantage of LOCK's type enforcement mechanism, the next step is to implement the design on LOCK. LOCK currently provides two interfaces for software development. For applications requiring no assurance, a fully functional Unix interface, LOCKix, is provided. Privileged applications, on the other hand, must be implemented on the TCB interface directly. A third interface that allows privileged applications to be developed on a LOCKix style interface is under development. It is described in Section 5.

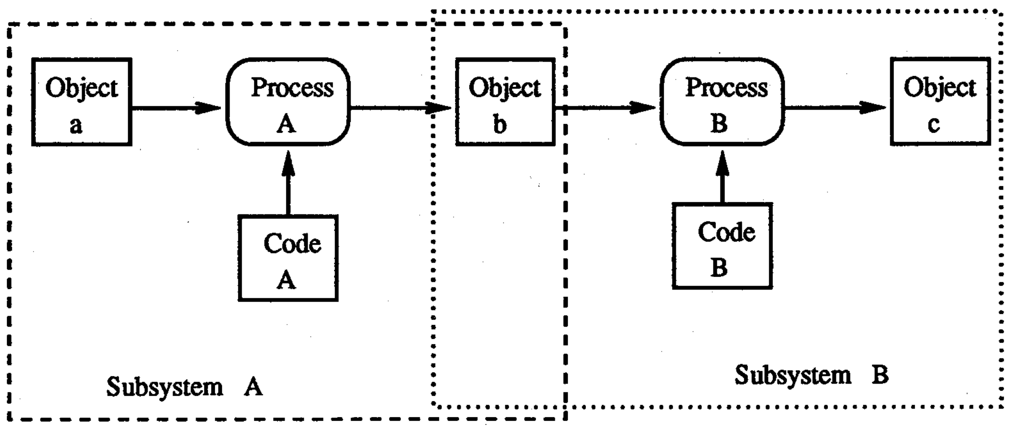


Figure 3: An Assured Pipeline. This example of an assured pipeline is composed of two filters, each designed and assured to perform its particular function. Process A filters the data in object a and places it into object b. Object b is a shared object used to construct the pipeline. Process B filters data from object b and places it into object c. An example of an assured pipeline might be a labeler process (process A) followed by a printer process (process B). Object c in this case would be the output from the printer. The labeler process would be assured to correctly label the data and put the labeled data in object b. The printer process would be assured to print the data it receives correctly.

4.1 Implementing Unprivileged LOCK Applications

For unprivileged software that does not need to communicate with other LOCK subjects, a developer can use the LOCKix programming environment. LOCKix is an unprivileged application providing a Unix interface on top of the LOCK TCB. It provides a fully functional single level Unix kernel with read only access to files at dominated levels. LOCKix is based on Unix System V, Release 1 and is over ninety percent system call compatible with Unix System V Release 2 as measured by the System V Verification Suite. The next release of LOCKix will be based on System V, Release 4.

LOCKix supports a C compiler, 68000 assembler, loader and C library. It also has program development utilities such as an archiver and "make", and runs many existing Unix programs with little or no modification. Most of the modifications required, in fact, are corrections of hardware dependent programming errors in older programs. Most modern Unix code ports reasonably easily. LOCKix provides a familiar programming interface and Unix library support.

The LOCK type enforcement mechanism allows a great deal of flexibility in controlling use of the LOCKix compiler. LOCK can be configured so that only users privileged to run LOCKix in a special domain can create objects that the LOCK host can execute. This prevents unauthorized users from creating LOCK executable code.

LOCKix currently does not support a debugger, but will at some time in the future. The present lack of a debugger makes LOCKix a less than ideal environment for program development. Further work is also needed to develop high level support for inter-subject communication. Multiple processes running inside the same LOCKix session communicate like any Unix process. However, no Unix library support currently exists to enable LOCKix processes to communicate with other LOCK subjects. LOCKix currently does not have a library interface to the LOCK TCB (although creation of one is planned), so the direct TCB calls currently required for inter-subject communication must be made in assembly language.

Because LOCKix presents a compatible Unix interface, the current development approach is for application developers to write, debug, test and run applications on their favorite Unix system, and then, once the application is ready for use, simply recompile and run it on LOCKix. This approach has been used with great success in porting Unix software to LOCKix. Portable software (Kermit, some GNU software, etc.) has been compiled and run on LOCKix without modification.

While the best method of implementing most unprivileged applications on LOCK is to develop them to run on LOCKix, there may be some applications that would require little or no support from LOCKix. Such applications could be implemented directly on the LOCK TCB interface used for privileged software. The method by which this is done is discussed in the following section.

4.2 Implementing Privileged LOCK Applications

Privileged software cannot depend on LOCKix as the underlying system because it is large and unassured, and if subverted, could cause the privileged software to be subverted also. Privileged software must be developed to run directly on the native LOCK TCB. The TCB provides a small set of well understood, well behaved primitives providing simple memory and communication facilities. The simplicity and power of the LOCK TCB interface makes development of sophisticated, multi-level assured applications possible.

Privileged (and some unprivileged) applications have been developed using the library interface to the TCB. A full set of routines for inter-subject communication, memory management, device handling, signaling and more are available in this library. A set of Unix stubs that simulates most of these library routines has been developed so that the first phase of debugging can take place on a Unix system, using its program development utilities. The LOCKix C compiler cannot be used to compile the code because it cannot generate the fully relocatable code required to run on the native TCB. LOCK TCB code is generated using a cross compiler. Once code is moved to the TCB, it must be integrated using a hardware level debugger. A software based debugging capability should be available some time in the future.

5 The Future of LOCK Software Development

In future LOCK systems the goal is to provide a complete software development environment in which both privileged and unprivileged software can be developed in the same manner and with the same ease. This section describes some of the ideas and enhancements that will make such an approach possible.

5.1 Features of the Software Development Environment

An Isolated Development Environment. The LOCK type enforcement mechanism can be used to create insulated test environments for development of privileged applications. By insulating the development and test environment, it becomes reasonable to develop and test privileged applications using LOCKix. For example, when testing a text downgrader a special LOCKix domain can be created that has read and trusted write access capabilities to a special type of test object. That domain would not be able to read or write any other type of object, and other domains would only be able to destroy that type of object. This allows controlled creation of a high level object that contains no high level information, which can be safely downgraded during testing. This way when testing the downgrade function, the domain restriction keeps any information from being accidentally or deliberately downgraded during testing. Less critical software can be developed in less insulated domains with fewer controls.

An Assurable Unix Interface Library. The LOCK assurable Unix interface library is a small subset of the Unix system call interface. This library will provide developers with access to a simple Unix file system, allowing them to specify object identifiers using a pathname. System calls providing file manipulation, interprocess and inter-subject communication, and signal management will also be provided.

This library will provide an interface for privileged software to be compiled using a subset of Unix system calls. The need for TCB interface stubs will be eliminated, and programming privileged software will be simplified due to the more familiar Unix interface.

Run-Time Environment Enhancements. The format and execution of TCB subjects will be changed to add support for non-relocatable program code. Also, the TCB subject calling conventions will be enhanced to provide arguments and environment information in a manner similar to a Unix system. With these enhancements, development of privileged software can take place in LOCKix without special support tools.

Complete Software Development Toolset. Another major improvement will be the addition of a standard Unix source level debugger to LOCKix. This will make it possible for developers to debug LOCKix code while running in LOCKix, and will complete the LOCKix programmer's toolset.

5.2 Using the Software Development Environment

With a complete toolset in place, a more Unix-like TCB run time environment, the assured Unix interface library, and the proper use of type enforcement, LOCKix will become an effective platform for developing both unprivileged and privileged LOCK applications. Developers will be able to write and test assured software in a special domain insulated from regular system users. In such an environment they can compile and run privileged (and unprivileged) code until they are satisfied with its correctness.

Applications would then be moved from a development environment to a production environment via an assured pipeline. The source code written in the assured software development environment would be of type *assurable code.* To compile this code in a format executable by general LOCK users, it would have to be reviewed using a privileged source code reviewer that runs in a domain that can read objects of type *assurable code* and can write objects of type *reviewed code.* The LOCKix compiler would then be run in a special domain that can read objects of type *reviewed code*, and write objects of a type that can be executed outside of the insulated development domain. Different review steps could be added or deleted as required by individual sites.

This model allows for controlled transition of software from development to operational status. It supports role separation, allowing sites to separate the roles of software developer, reviewer and installer. It uses many of the features of type enforcement to provide a secure, controlled environment for the complete LOCK application development cycle.

6 Examples

In order to illustrate the manner in which critical applications can be designed and implemented on LOCK, we present some examples.

Example 1. A Privileged Subsystem.

For our first example consider a subsystem that is designed to run as a single privileged subject on top of a TCB. Such a subsystem might be a multilevel DBMS that performs all of its processing at system high and then downgrades the results.

If the design is such that the entire DBMS cannot be easily decomposed into modules, some of which need to be privileged and others that do not, then the full advantage of type enforcement cannot be gained. However, it is still desirable to create a special DBMS domain, in which the DBMS privileged subject would run, and special types for the DBMS files. It would then be possible to run the DBMS as an isolated subject on the system as described in figure 1.

Although implementing the subsystem directly on a version of the standard LOCKix system might be very easy, this approach has the disadvantage that since LOCKix is unprivileged, if it is corrupted, the privileged subsystem might be subverted. This danger can be mitigated by using a small, well-understood LOCKix subset, that only supports the functionality required by the DBMS, on which to perform the port and then configuring the DDT so that this code object can not be modified and so that it is the only code object that can be executed from the special DBMS domain. Note that if the application is properly isolated, even if it is subverted, it can only affect information available within its subsystem.

Of course, the danger that the LOCKix subsystem can be subverted is always present. To insure that the subsystem cannot be compromised, it would be necessary to implement it using either the LOCK TCB interface or, in the future, the assurable Unix interface. If it provides the required support that the privileged subsystem needs, then the assurable Unix interface is probably the best choice, since the implementation would be easier. In fact, if minimal additional support is required, it might be desirable to add this support in an assured manner. In this way additional functionality can be added to the assurable Unix interface in an incremental manner.

Example 2. A Modularized Subsystem

The real advantage of type enforcement is only obtained when a subsystem is designed so that its security and integrity components are separated into modules which are small enough so that the corresponding code can be properly assured for correctness. This highly reliable code, and the data it deals with, can then be isolated using special domains and types and the remainder of the subsystem can be implemented as an unprivileged subject.

As an example consider a multilevel DBMS design, such as LDV [9], [10] in which all privileged processing can be isolated in a few modules, and most of the DBMS functionality is unprivileged. The unprivileged portion of the DBMS could be implemented on standard LOCKix using standard database code. This might involve nothing more than compiling the code on LOCKix. The privileged portions would be implemented as discussed in the previous example, using either the LOCK TCB or the assurable Unix kernel. The LDV design is an example of an assured pipeline, since any query first passes through the unprivileged DBMS, then the privileged filter that determines what information can be released, and then the response passes back out through the unprivileged DBMS.

Example 3. A Role Based Subsystem

Examples 1 and 2 illustrate the manner in which unprivileged and privileged software can be ported to LOCK to take advantage of LOCKix and type enforcement. To illustrate how roles can be implemented, consider a simple example in which a DBMS is used to create reports which must be reviewed by a human before they are released. The DBMS may itself be unprivileged, but the previewer subject that handles the review processing is privileged to correctly display the report, so that a user can review it, and only release it if it passes review. In effect, the previewer acts as a filter. Furthermore, the role of the reviewer is only allowed to certain privileged individuals.

This subsystem can be implemented on LOCK in much the same way as described in Example 2 with the previewer being put in a special domain that acts as a filter between objects of type *report* and objects of type *reviewed report*. The role of the reviewer is then implemented by using the Role Authorization Table. Only users who are allowed to be reviewers are permitted to have subjects that execute in the previewer domain. Hence, only these users are allowed to perform the role of a reviewer.

7 Conclusion

This paper has discussed some issues involved in designing and implementing an application on the LOCK system, and some of the features LOCK provides to aid application development.

LOCK's type enforcement mechanism allows an application designer to decompose an application into modules which can then be separated into separate domains and types allowing the interaction between the modules to be strictly controlled. Type enforcement also allows separation of duties, simplified trust management, creation of assured pipelines and role enforcement.

Once an application is designed, LOCKix and the TCB interface provide tools a developer can use to build privileged and unprivileged applications. In the future, LOCKix will add functionality to its software development environment, and the assurable Unix interface will allow privileged modules to be implemented in an even more efficient manner.

References

[1] "Trusted Computer System Evaluation Criteria", Department of Defense Standard 5200.28-STD, December 26, 1985.

[2] O. Sami Saydjari, J. Beckman and J. Leaman, "LOCKing Computers Securely", *Proceedings of the 10th DoD/NBS Computer Security Conference*, NBS, 1987, pp. 129-140.

[3] W.E. Boebert, "Constructing an Infosec System Using the LOCK Technology", *Proceedings of the 11th National Computer Security Conference*, NBS, 1988, Postscript Volume.

[4] O. Sami Saydjari, J. Beckman and J. Leaman, "LOCK Trek: Navigating Uncharted Space", *Proceedings of the IEEE Symposium on Security and Privacy*, 1989, pp.167-175.

[5] W.E. Boebert and R.Y. Kain, "A Practical Alternative to Heirarchical Integrity Policies", *Proceedings of the 8th National Computer Security Conference*, NBS, 1985, pp. 18-27.

[6] W.E. Boebert and Steven M. Miller, "Comparison of Mechanisms for Implementing Protected Software Subsystems", unpublished draft.

[7] D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proceedings of the 1987 Symposium on Security and Privacy*, 1987, pp. 184-194.

[8] D. J. Thomsen and J. T. Haigh, "A Comparison of Type Enforcement and Unix Setuid Implementation of Well Formed Transactions," *Proceedings of the 1990 Computer Security Applications Conference*, 1990, pp. 304-312.

[9] J.T. Haigh, R.C. O'Brien, and D. Thomsen, "The LDV Secure Relational DBMS Model", *Proceedings of the 1990 IFIP Database Workshop*, to appear.

[10] P.D. Stachour and B. Thuraisingham, "Design of LDV: A Multilevel Secure Relational Database Management System", *IEEE Transactions on Knowledge and Data Engineering*, 1990, Vol.2, pp. 190-209.

THE DEVELOPMENT OF A LOW-TO-HIGH GUARD

Michelle J. Gosselin
The MITRE Corporation
Burlington Rd., Bedford, MA 01730

ABSTRACT

This report details the development of a guard to monitor electronic traffic between two computer systems. The guard is intended to operate between two computer systems that are accredited to operate at different security levels. One system (the high system) must be accredited to process all information on the other system (the low system). The purpose of the guard is to automate the delivery of information from the low system to the high system while preventing any flow of information from the high system to the low system.

ACKNOWLEDGEMENT

This paper is based on work done for the Air Force, ESD-ICD. Their support in this effort is greatly appreciated. Many co-workers at the MITRE Corporation are also to be thanked for their contributions to this effort. Specifically, Karen Johnson is thanked for implementing major pieces of this design.

INTRODUCTION

The low-to-high security guard described in this report was developed for two specific systems accredited at two specific security levels. However, the design of the guard allows for the guard to be easily applied to other environments with systems operating at different security levels involving a low-to-high information flow.

The purpose of the guard is to automate the delivery of data from the low system to the high system while preventing any flow of information from the high system to the low system.

The development of the guard occured in three phases. The first phase involved the selection of the guard platform. Once the platform was chosen, the operational concept of the guard was defined. The third and final phase was the actual implementation of the operational concept. Each of these phases is discussed in detail in this report.

Also discussed in this report are the issues that must be addressed before the guard can be accredited for operation.

GUARD PLATFORM

SELECTION

The first task in developing the guard was to select a suitable hardware and software platform. There were three requirements that were considered when selecting the guard platform. These requirements are stated in the following list.

1. The guard should be a low cost system with a maximum cost near $10,000.

2. Because the guard processes sensitive data, a guard must satisfy certain security requirements mandated by the Department of Defense (DOD) in DOD Directive

5200.28. According to Directive 5200.28, a system that processes classified information and that requires controlled access protection must meet, at a minimum, the C2 class of security requirements specified in the DOD Trusted Computer System Evaluation Criteria (TCSEC). The operating mode and the level of trust that is required for the guard is determined by risk assessment which is determined by the minimum clearance of the users and the maximum data sensitivity. For systems of different security levels, B class services that provide mandatory access control and can separate different security levels are required. The directives that govern the interfacing of intelligence systems with nonintelligence systems generally require that the guards be multilevel and provide a B2 level of trust as a minimum.

3. Since the purpose of the guard is to prevent the transmission of data from high to low, the guard must be capable of keeping any network traffic intended for the low system separate from any network traffic intended for the high system. One method of keeping the traffic separate is for the guard to use separate network protocol stacks and Ethernet cards to interface with each system. The configuration of such a system is displayed in figure 1.

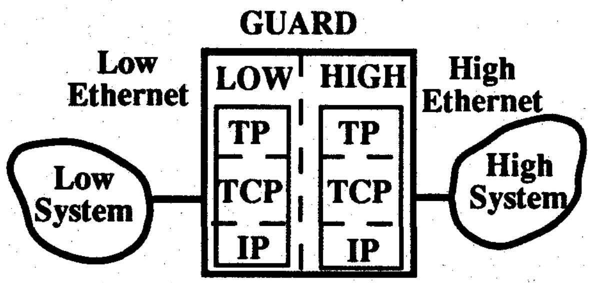


Figure 1. Configuration With Separate Ethernets

There are several systems that meet or surpass the C2 class of security requirements. However, some of these systems far exceed the cost limit of $10,000, and many of the systems do not have separate protocol stacks. The only system that meets the above criteria is the Trusted Xenix operating system produced by Trusted Information Systems (TIS). Trusted Xenix has received a B2 rating, which surpasses the required C2 rating. The operating system runs on a variety of hardware platforms, including many INTEL 80286 and 80386 based workstations. Trusted Xenix and an appropriate hardware platform can be purchased for approximately $10,000.

TIS has developed a version of Trusted Xenix that implements two separate instantiations of the networking protocols. Two separate Ethernet cards (and ports) are also supported. Traffic can be kept separate by regulating the access to the protocols and to the ports. The method in which access is regulated is explained in detail in the following section.[1]

SECURITY POLICY

Trusted Xenix is a Unix-based operating system. The operating system has several types of objects to which information can be written. These objects include files, directories, and

[1] Unfortunately, the network interfaces are not included in the evaluated configuration. This matter will have to be addressed prior to or during the accreditation process.

ports. A directory can contain files and other directories. A port can be used to transmit information to devices external to the workstation.

Actions on the operating system objects are performed by processes. For instance, a process can read information from and write information to a file, a directory, or a port. A process can also initiate other processes. An example of a process is the networking process, referred to as the *inet* daemon, that handles information coming into the system from the network. On the guard, there are two *inet* daemons; one for the low side and one for the high side.

In the Trusted Xenix environment, each object and process has a sensitivity label associated with it. The sensitivity label represents the sensitivity of the data contained in the object or process. A sensitivity label is composed of two pieces; a hierarchical classification and a set of nonhierarchical categories. A sensitivity label S1 dominates a sensitivity label S2 if the classification of S1 is higher than or equal to the classification of S2 and the category set of S2 is a subset of the category set of S1.

The security policy of Trusted Xenix is composed of two pieces; a read policy and a write policy. The read policy states that a process can read an object if the sensitivity label of the process dominates the sensitivity label of the object. The write policy states that a process can write to an object (which includes creation and deletion) if the sensitivity label of the process equals the sensitivity label of the object. Trusted Xenix enforces strict adherence to these policies. For a process to override these policies, the process must possess special privileges granted by the System Security Officer using mechanisms provided by Trusted Xenix.

The security policy of Trusted Xenix also applies to the unprivileged processes associated with the guard software. The unprivileged processes of the guard software do not have a separate security policy, and are forced to follow the policy established by Trusted Xenix. However, there is one privileged guard process that does violate the Trusted Xenix security policy. This process is privileged to append one byte of information to a file that exists in a directory that is at a lower security level than the level of the process[2]. The process does not otherwise violate the Trusted Xenix security policy.

OPERATING CONCEPT

Before discussing the details of the guard software, it is necessary to understand the operating concept of the guard. The operating concept is based on the following scenario: the low system, which is accredited to process SECRET A data, sends a database update to the high system, which is accredited to process SECRET A/B data. The high system then provides either an acknowledgement or a negative acknowledgement of the receipt of the database update. The operating concept can be separated into two phases. Phase I involves the transfer of the database update from low to high. Phase II involves the transfer from high to low of an indication of either having received (an acknowledgement) or not received (a negative acknowledgement) the database update. Phase I of the operating concept is depicted in figure 2, and phase II is shown in figure 3. In both figures, processes are represented by circles, and files (the database update and the acknowledgment or negative acknowledgement) are represented by squares.

2 The privileged process is discussed in detail in the paper.

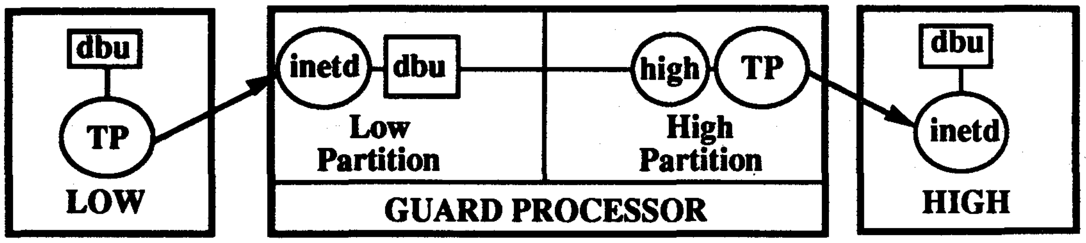


Figure 2. Operating Concept for File Transfer

The operational sequence of transferring the file is as follows. An authorized user of the low system will initiate a file transfer process using an application layer transfer protocol (TP), such as Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP). The *inet* daemon (*inetd*) on the low side of the guard will download the database update (*dbu*), in the form of a file, to the guard platform in the low partition (or directory).

Once the dbu file is stored on the low partition, the guard software performs various actions (such as reading and copying the file) in order to transfer the file to the high system. The guard software is separated into three different processes: a low process, a guard process, and a high process. The first process that takes any action on the file is the high process. When the high process detects the presence of the file in the low partition by reading the contents of the low partition, the high process initiates a TP connection with the high system. Once the connection has been established, the file is transferred across the TP connection to the high system.

After the file has been successfully transferred, an acknowledgment of the successful transfer is sent back to the low system from the high system via the guard software. If the file was not successfully transferred, a negative acknowledgment is sent back to the low system from the high system via the guard software. The operating concept of the transfer of the acknowledgment (ack) or negative acknowledgment (nack) is depicted in figure 3.

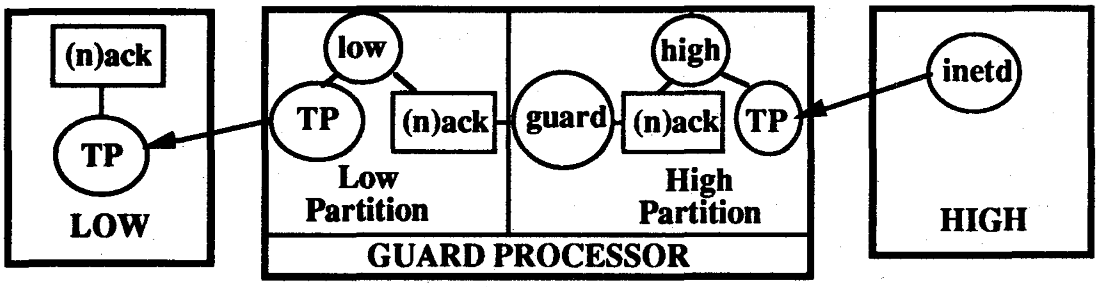


Figure 3. Operating Concept for Transfer of (N)ack

The operational sequence of transferring the (n)ack is as follows. The high process determines from the information received from the high *inet* daemon whether or not the dbu file transfer was successful. If successful, the high process creates an acknowledgment, in the form of a file, of the successful transfer on the high partition. If not successful, the high process creates a negative acknowledgement, in the form of a file, to indicate the failure on the high partition.

The guard process, after detecting the presence of the (n)ack file on the high partition creates another (n)ack on the low partition by appending information to the original dbu file. Once the acknowledgment has been created, the guard process deletes the original (n)ack file on the high partition.

Finally, the low process detects that the size of the (n)ack file (previous dbu file) has increased and transmits the (n)ack file to the low system. The low process then deletes the (n)ack file on the low partition.

IMPLEMENTATION

The guard software is composed of three continuously running processes; a low process, a guard process, and a high process. Each of these processes is automatically started when the system is started. Each process is cyclical in that it executes a sequence of steps and then returns to the first step. The first step for each process is to detect the presence of a particular file. If the file is not detected, the process temporarily suspends execution for a configurable period of time (currently ten seconds). After this delay, the process again starts at the first step. The low process runs at the SECRET A level, and both the guard process and the high process run at the SECRET A/B level. The guard software also works in conjunction with an inet daemon dedicated to the low side and an inet daemon dedicated to the high side. Each of these processes is described in detail in this section.

TRANSFER OF THE DBU

Figure 4 depicts the actions that are taken on the incoming dbu file. Also provided in the figure is the resulting sensitivity labels on both the dbu file (F) and the process (P) involved in the action.

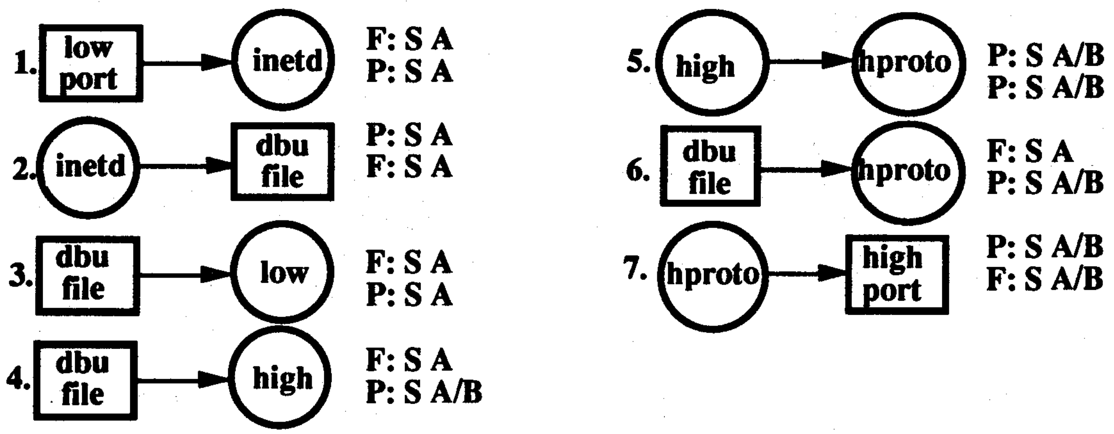


Figure 4. Transfer of the DBU

The dbu file arrives in the following manner on the system. The low user sends a file to the guard via a transport protocol. The only naming convention that the low user must follow is to not begin the name of the file with the character ".". When a low user sends data to the guard, the data comes across the low Ethernet and through the port dedicated to that Ethernet. This port is labeled SECRET A. When data comes across this port, the low inet daemon, which is labeled SECRET A, reads the data.[3] The *inet* daemon then writes the data to a file

[3] As stated previously, a port is an object that contains data. The data that the object contains is the data that is coming across the port. Therefore, the data coming across the port has the same sensitivity label that the port does.

in the low directory which is labeled SECRET A. This file is created at the SECRET A level.

The low process checks the low directory for a file containing a database update. The search of the low directory is allowed since both the low process and the low directory are SECRET A. If a database update (dbu) file is detected, the low process records the size of the dbu file.

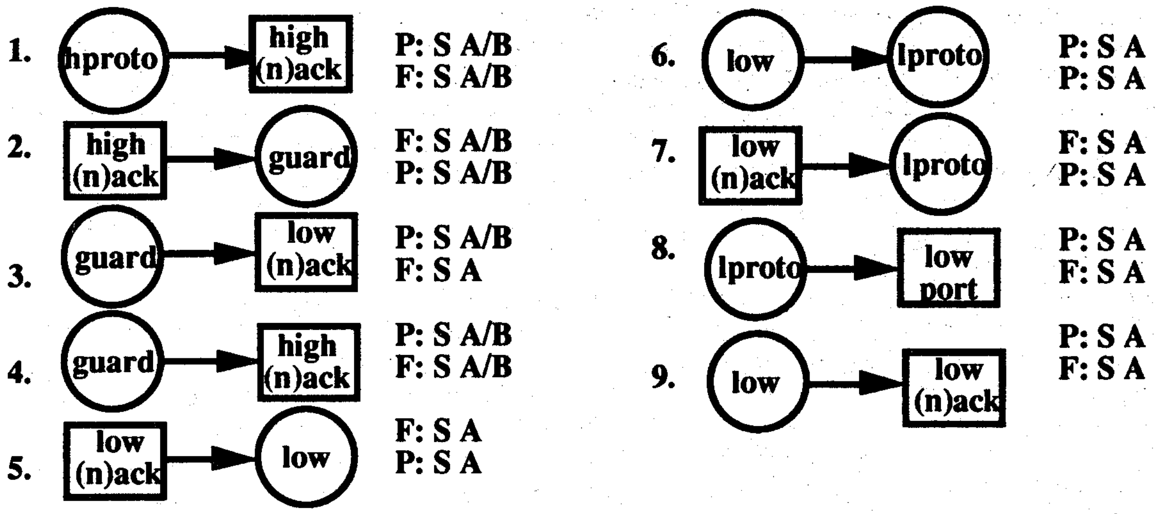
The high process also checks the low directory for a file containing a database update. The search of the low directory is allowed since the high process is SECRET A/B and the low directory is SECRET A. If there is a dbu file, the high process initiates a new process that executes the commands found in a script file called *hproto* (for high protocol).

The *hproto* script file contains a list of executable commands. This script file is entirely tailorable to the specific environment that is hosting the guard. The script can be changed without having to recompile any of the guard software. This allows for complete freedom in choosing any of the TCP/IP based protocols to be used for file transfers. This includes FTP, SMTP, and rcp. Currently, the guard software calls a version of *hproto* that uses FTP as the transfer protocol. Hproto issues the appropriate TP command in order to transmit the dbu to high using the protocol stack dedicated to the high side.

After the transfer of the dbu file, the creation and transfer of the (n)ack file takes place as described in the following section.

TRANSFER OF THE (N)ACK

Figure 5 depicts the actions that are taken on the (n)ack file. Also provided in the figure is the resulting sensitivity labels on both the file (F) and the process (P) involved in the action.



1. hproto → high (n)ack P: S A/B F: S A/B
2. high (n)ack → guard F: S A/B P: S A/B
3. guard → low (n)ack P: S A/B F: S A
4. guard → high (n)ack P: S A/B F: S A/B
5. low (n)ack → low F: S A P: S A
6. low → lproto P: S A P: S A
7. low (n)ack → lproto F: S A P: S A
8. lproto → low port P: S A F: S A
9. low → low (n)ack P: S A F: S A

Figure 5. Transfer of the (N)ack

As the dbu file is being transfered, the data sent back from the inet daemon on the high host is then analyzed by *hproto* to determine if the file was successfully sent. If the transfer was successful, an ack file is created in the high directory with the name of the dbu file in the low directory. This ack file contains one byte that has a value of 0.

If the transfer was not successful, attempts are made to retransmit the file for a configurable number of times. If the maximum number of retransmission attempts is reached, the high process creates a nack file in the high directory with the name of the dbu file in the low directory. The nack file contains one byte that has a value of 1.

Once an (n)ack has been received, the next stage of the (n)ack transfer cycle involves the guard process.

The guard process periodically checks for (n)acks in the high directory. The guard process determines that a (n)ack exists when there is a file with a size of one byte. If a file is greater than one byte in length, a violation has occured, and the violation is audited[4].

Upon detection of a (n)ack, the guard process attempts to open a file in the low directory that has the same name as the (n)ack. If no such file exists, a (n)ack file exists that does not have any corresponding dbu file. Therefore, a violation is indicated, and the violation is audited.

If the guard process finds a corresponding dbu file, the guard process reads the byte of information in the (n)ack file in the high directory. If the byte is a 0 (ack) or a 1 (nack), the high process appends the byte to the dbu file in the low directory. This appended dbu file is now considered the low ack file. The guard process then deletes the (n)ack in the high directory. The writing of information to the SECRET A dbu file by the SECRET A/B guard process requires a special security privilege granted by the operating system to override the write policy.

If the byte in the high (n)ack file holds a value of other than a 0 or a 1, a violation is indicated, and the violation is audited. The (n)ack in the high directory is deleted.

The next stage of the database and (n)ack transfer cycle involves the low process.

The low process periodically checks the low directory for a file containing a (n)ack. The low process determines that a (n)ack exists when the size of a file has increased by one byte. If a (n)ack is found, the low process reads the file, which is labeled SECRET A, for the additional byte of information. The low process then deletes all previous information from the file. Once the low process has modified the (n)ack file to contain only the byte of information from the guard process, the low process initiates a new process that executes the commands found in a script file called *lproto* (for low protocol).

The *lproto* script file contains a list of executable commands. This script file is entirely tailorable to the specific environment that is hosting the guard. The script can be changed without having to recompile any of the guard software. This allows for complete freedom in choosing any of the TCP/IP based protocols to be used for file transfers. This includes FTP, SMTP, and rcp. Currently, the low process calls a version of *lproto* that uses FTP as the transfer protocol. *Lproto* issues the appropriate TP command in order to transmit the (n)ack to low using the protocol stack dedicated to the low side. low must determine if the file is an ack or nack based on the contents of the file.

Once the commands in the *lproto* script file have been executed, the low process deletes the the acknowledgment file.

4 All audit records includes the date and time when the violation was detected, the name of the (n)ack file, and the contents of the (n)ack file.

ACCREDITATION

For the low/high guard to be used operationally, it must first be certified and accredited. This section discusses the issues that need to be addressed in certifying and accrediting the guard.

ACCREDITATION AND CERTIFICATION PLAN

To properly accredit the low/high guard for operation, an accreditation and certification plan should be written for the guard. The primary purpose of an accreditation and certification plan is to serve as a handbook for the Information System Security Officer (ISSO) and the Designated Approving Authority (DAA) in carrying out their roles in the accreditation and certification process. An accreditation and certification plan usually provides the following information:

1. A system overview describing the operational environment,
2. Security requirements the system must meet,
3. Documentation that must be supplied to the ISSO and/or DAA,
4. Organizational responsibilities, and
5. A detailed description of the accreditation and certification process.

EVALUATION

As stated previously, the guard base (Trusted Xenix on a 286 or 386 machine) has received a B2 rating from the National Computer Security Center (NCSC). However, changes that have been made to this base to implement the guard effect this rating. Both the networking software and the guard software have to be analyzed for their effect on the overall rating.

Networking Software

NCSC currently does not evaluate any networking software and accompanying hardware. Each system under evaluation is treated as a standalone system. Therefore, the networking software used by the guard may not meet the NCSC B2 security requirements. During the accreditation and certification process, the networking software resident in the kernel of the operating system would have to be inspected by the ISSO and the DAA in order to determine whether or not the software was trustworthy enough for the environment. The networking software would have to analyzed for covert channels and for its effect on the trusted operating system. According to TIS, there are two separate protocol stacks that can be labeled separately. These separate protocol stacks and labels allow the separation between low and high to be maintained. The ISSO and DAA would have to verify that there are indeed two separate stacks.

Guard Software

Since the guard software is given the privilege to violate the security policy of the operating system when creating the acknowledgment in the low directory, this software must also be inspected. However, inspecting the guard software is a much easier task than inspecting the networking software since the privileged portion of the guard software consists of six lines of code. Inside the guard software, the following steps are taken:

1. A privilege to override the mandatory access control policy is granted.
2. A file in the low directory is opened for writing.

3. A byte of information, containing either a 0 or a 1, is appended to the file.
4. The file is closed.
5. The privilege to override the mandatory access control policy is revoked.

These five lines of code could quickly be analyzed with respect to the B2 requirements, if the desire is to maintain a B2 system. One of the B2 requirements that may pose a problem is the covert channel analysis requirement. However, if the action of creating an acknowledgment file does create a covert channel, the channel could easily be reduced to an acceptable size by limiting the rate at which acknowledgments are created by the guard process.

OPERATIONAL ENVIRONMENT

The operational environment of the guard must also be considered when accrediting the guard. Aspects of the environment that must be considered are the location, additional uses of the guard, and the users.

Location

The guard should be located in a facility where access is restricted to individuals who are allowed to process SECRET A/B data.

Guard Uses

The guard is intended strictly for use as a guard. No application packages, such as word processors and spread sheets, are resident on the guard. Without any application packages, the desire to use the guard for other purposes will be minimal. By limiting the use of the guard, the number of accidental security infractions will be limited.

Users

There are two groups of users that have accounts on the guard. One group consists of the security personnel who are responsible for maintaining the system. The security personnel include a security administrator, an auditor, and an operator. The security administrator maintains user accounts and is responsible for the security of the system. The auditor analyzes the audit trail, and the operator performs day-to-day operations, such as systems backups.

The other group of users are general users who do not have system responsibilities. There is one general user with an account on the guard, which is the guard user. The guard user is the owner of the low process, the guard process, and the high process respectively. Since these processes are automatically started when the system is turned on, there is no need for this user to log onto the system. Therefore, for security purposes, the guard user should be administratively prevented from logging in.

By limiting the number of users of the guard, the number of intentional security infractions will be limited.

CONCLUSION

The guard documented in this paper provides for the automated transfer of a database update from a low system to a high system. The guard also automatically relays an acknowledgment of a successful transfer or a negative acknowledgement is the transfer was not successful back to the low system.

The guard also prevents the flow of information from the high community to the low community. There are two types of the high-to-low information flow; the high side writing to the low side, and the low side reading from the high side. The way in which these flows are prevented is as follows. For a user of the high side to gain access to the low side, the user must first gain access to the guard. Access can only be gained through the Ethernet card on the high side. The port associated with this Ethernet card is labeled SECRET A/B. Therefore, the operating system will automatically label any data coming through that port with the SECRET A/B label. Similarly, a user on the low side must first access the guard via the low side Ethernet card before accessing the high side. Since the port associated with the low side Ethernet card is labeled SECRET A, all data coming through that port will be labeled SECRET A by the operating system. If a SECRET A/B process running on behalf of a user from the high side attempted to write to the low side, the operating system would disallow the write because the low side port is only SECRET A. If a SECRET A process running on behalf of a user from the low side attempted to read from the high side, the operating system would disallow this since the high side port is SECRET A/B.

There is one instance where an information flow from high to low is allowed by the guard. This capability is granted to the guard process through a privilege mechanism. The guard process is trusted to append one byte of information to an existing file in the low directory. The byte contains a value of either 0 or 1. The value of the byte is determined from information supplied from high, and this value is passed to the low system. The guard process is trusted to append strictly one byte containing no other value except 0 or 1. The guard process cannot create a new file in the low directory.

As stated previously, the guard must be formally accredited before it is used operationally. It might also be useful to make further enhancements to the guard before employing it. For instance, the auditing and report generation features of the guard could be specialized for each specific environment. Trusted Xenix is responsible for auditing system events and creating an audit trail of these events. However, the Trusted Xenix auditing capabilities are general to the overall system. Auditing capabilities could be developed that are more specific to the guard operations. This could reduce the size of the audit trail and make the audit trail easier to understand.

Other features that could be added are host authentication and error reporting. Host authentication would be used to verify both the low host and the high host as valid members of the networks. Error reporting would give a better indication to the low host as to the condition of the message when it arrived at the high host.

DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and An Early Prototype

Steven R. Snapp[1], *James Brentano*[2], *Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha*[1], *Tim Grance*[3], *Daniel M. Teal*[3], *and Doug Mansur*[4]

Computer Security Laboratory
Division of Computer Science
University of California, Davis
Davis, California 95616

ABSTRACT

Intrusion detection is the problem of identifying unauthorized use, misuse, and abuse of computer systems by both system insiders and external penetrators. The proliferation of heterogeneous computer networks provides additional implications for the intrusion detection problem. Namely, the increased connectivity of computer systems gives greater access to outsiders, and makes it easier for intruders to avoid detection. IDS's are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user. We are designing and implementing a prototype Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction (through individual host and LAN monitors) with centralized data analysis (through the DIDS director) to monitor a heterogeneous network of computers. This approach is unique among current IDS's. A main problem considered in this paper is the Network-user Identification problem, which is concerned with tracking a user moving across the network, possibly with a new user-id on each computer. Initial system prototypes have provided quite favorable results on this problem and the detection of attacks on a network. This paper provides an overview of the motivation behind DIDS, the system architecture and capabilities, and a discussion of the early prototype.

1. Introduction

Intrusion detection is defined to be the problem of identifying individuals who are using a computer system without authorization (i.e., *crackers*) and those who have legitimate access to the system but are exceeding their privileges (i.e., the *insider threat*). Work is being done elsewhere on Intrusion Detection Systems (IDS's) for a single host [8, 10, 11] and for several hosts connected by a network [6, 7, 12]. Our own earlier work on the Network Security Monitor (NSM) concentrated on monitoring a broadcast Local Area Network (LAN) [3].

The proliferation of heterogeneous computer networks has serious implications for the intrusion detection problem. Foremost among these implications is the increased opportunity for unauthorized access that is provided by the network's connectivity. This problem is exacerbated when dial-up or internetwork access is allowed, as well as when unmonitored hosts (viz. hosts without audit trails) are present. The use of distributed rather than centralized computing resources also implies reduced control over those resources. Moreover, multiple independent computers are likely to generate more audit data than a single computer, and this audit data is dispersed among the various systems. Clearly, not all of the audit data can be forwarded to a single IDS for analysis; some analysis must be accomplished locally.

[1] Haystack Laboratories, Inc., 8920 Business Park Dr, Suite 270, Austin, TX 78759
[2] Pacific Gas and Electric Company, 77 Beale St, Room 1871B, San Francisco, CA 94106
[3] United States Air Force Cryptologic Support Center, San Antonio, TX 78243
[4] Lawrence Livermore National Labs, Livermore, CA 94550

This paper describes a prototype Distributed Intrusion Detection System (DIDS) which generalizes the target environment in order to monitor multiple hosts connected via a network as well as the network itself. The DIDS components include the DIDS director, a single host monitor per host, and a single LAN monitor for each LAN segment of the monitored network. The information gathered by these distributed components is transported to, and analyzed at, a central location (viz. an expert system, which is a sub-component of the director), thus providing the capability to aggregate information from different sources. We can cope with any audit trail format as long as the events of interest are provided.

DIDS is designed to operate in a heterogeneous environment composed of C2 [1] or higher rated computers. The current target environment consists of several hosts connected by a broadcast LAN segment (presently an Ethernet, see Fig. 1). The use of C2-rated systems implies a consistency in the content of the system audit trails. This allows us to develop standard representations into which we can map audit data from UNIX, VMS, or any other system with C2 auditing capabilities. The C2 rating also guarantees, as part of the Trusted Computing Base (TCB), the security and integrity of the host's audit records. Although the hosts must comply with the C2 specifications in order to be monitored directly, the network related activity of non-compliant hosts can be monitored via the LAN monitor. Since all attacks that utilize the network for system access will pass through the LAN segment, the LAN monitor will be able to monitor all of this traffic.

Section 2 motivates our work by describing the type of behavior which DIDS is intended to detect. In Section 3 we present an overview of the DIDS architecture. In Section 4 we formulate the concept of the network-user identification (NID), an identifier for a network-wide user, and describe its use in distributed intrusion detection. Sections 5 and 6 deal with the host and LAN monitors, respectively, while Section 7 discusses the expert system and its processing mechanisms based on the NID. Section 8 provides some concluding remarks.

2. Scenarios

The detection of certain attacks against a networked system of computers requires information from multiple sources. A simple example of such an attack is the so-called *doorknob* attack. In a doorknob attack the intruder's goal is to discover, and gain access to, insufficiently-protected hosts on a system. The intruder generally tries a few common account and password combinations on each of a number of computers. These simple attacks can be remarkably successful [4]. As a case in point, UC Davis' NSM recently observed an attacker of this type gaining super-user access to an external computer which did not require a password for the super-user account. In this case, the intruder used *telnet* to make the connection from a university computer system, and then repeatedly tried to gain access to several different computers at the external site. In cases like these, the intruder tries only a few logins on each machine (usually with different account names), which means that an IDS on each host may not flag the attack. Even if the behavior is recognized as an attack on the individual host, current IDS's are generally unable to correlate reports from multiple hosts; thus they cannot recognize the *doorknob* attack as such. Because DIDS aggregates and correlates data from multiple hosts and the network, it is in a position to recognize the doorknob attack by detecting the pattern of repeated failed logins even though there may be too few on a single host to alert that host's monitor.

In another incident, our NSM recently observed an intruder gaining access to a computer using a guest account which did not require a password. Once the attacker had access to the system, he exhibited behavior which would have alerted most existing IDS's (e.g., changing passwords and failed events). In an incident such as this, DIDS would not only report the attack, but may also be able to identify the source of the attack. That is, while most IDS's would report the occurrence of an incident involving user "guest" on the target machine, DIDS would also report that user "guest" was really, for example, user "smith" on the source machine, assuming that the source machine was in the monitored domain. It may also be possible to go even further back and identify all of the different user accounts in the "chain" to find the initial launching point of the attack.

Another possible scenario is what we call *network browsing*. This occurs when a (network) user is looking through a number of files on several different computers within a short period of time. The browsing activity level on any single host may not be sufficiently high enough to raise any alarm by itself. However, the network-wide, aggregated browsing activity level may be high enough to raise suspicion on this user. Network browsing can be detected as follows. Each host monitor will report that a particular user is browsing on that system, even if the corresponding degree of browsing is small. The expert system can then aggregate such information from multiple hosts to determine that all of the browsing activity corresponds to the same network user. This scenario presents a key challenge for DIDS: the tradeoff between sending all audit records to the director versus missing attacks because thresholds on each host are not exceeded.

In addition to the specific scenarios outlined above, there are a number of general ways that an intruder can use the connectivity of the network to hide his trail and to enhance his effectiveness. Some of the attack configurations which have been hypothesized include *chain* and *parallel* attacks [2]. DIDS combats these inherent vulnerabilities of the network by using the very same connectivity to help track and detect the intruder. Note that DIDS should be at least as effective as host-based IDS's (if we implement all of their functionality in the DIDS host monitor), and at least as effective as the stand-alone NSM.

3. DIDS Architecture

The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDS's. The components of DIDS are the *DIDS director*, a single *host monitor* per host. and a single *LAN monitor* for each broadcast LAN segment in the monitored network. DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts. The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation. Reports are sent independently and asynchronously from the host and LAN monitors to the DIDS director through a communications infrastructure (Fig. 2). High level communication protocols between the components are based on the ISO Common Management Information Protocol (CMIP) recommendations, allowing for future inclusion of CMIP management tools as they become useful. The architecture also provides for bidirectional communication between the DIDS director and any monitor in the configuration. This communication consists primarily of notable events and anomaly reports from the monitors. The director can also make requests for more detailed information from the distributed monitors via a "GET" directive, and issue commands to have the distributed monitors modify their monitoring capabilities via a "SET" directive. A large amount of low level filtering and some analysis is performed by the host monitor to minimize the use of network bandwidth in passing evidence to the director.

The host monitor consists of a *host event generator* (HEG) and a *host agent*. The HEG collects and analyzes audit records from the host's operating system. The audit records are scanned for *notable events*, which are transactions that are of interest independent of any other records. These include, among others, failed events, user authentications, changes to the security state of the system, and any network access such as *rlogin* and *rsh*. These notable events are then sent to the director for further analysis. In enhancements under development, the HEG will also track user sessions and report anomalous behavior aggregated over time through user/group profiles and the integration of Haystack [10] into DIDS. The host agent handles all communications between the host monitor and the DIDS director.

Like the host monitor, the LAN monitor consists of a *LAN event generator* (LEG) and a *LAN agent*. The LEG is currently a subset of UC Davis' NSM [3]. Its main responsibility is to observe all of the traffic on its segment of the LAN to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on such network activity as *rlogin* and *telnet* connections, the use of security-related services, and changes in network traffic patterns.

The DIDS director consists of three major components that are all located on the same dedicated workstation. Because the components are logically independent processes, they could be distributed as well. The *communications manager* is responsible for the transfer of data between the director and each of the host and the LAN monitors. It accepts the notable event records from each of the host and LAN monitors and sends them to the *expert system*. On behalf of the expert system or user interface, it is also able to send requests to the host and LAN monitors for more information regarding a particular subject. The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole. The expert system is a rule-based system with simple learning capabilities. The director's *user interface* allows the System Security Officer (SSO) interactive access to the entire system. The SSO is able to watch activities on each host, watch network traffic (by setting "wire-taps"), and request more specific types of information from the monitors.

We anticipate that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS. This will give the SSO the ability to actively respond to attacks against the system in real-time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.

4. The Network-user Identification (NID)

One of the more interesting challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and tracking an intruder to their source. In a networked environment, an intruder may often choose to employ the interconnectivity of the computers to hide his true identity and location. It may be that a single intruder uses multiple accounts to launch an attack, and that the behavior can be recognized as suspicious only if one knows that all of the activity emanates from a single source. For example, it is not particularly noteworthy if a user inquires about who is using a particular computer (e.g., using the UNIX *who* or *finger* command). However, it may be indicative of an attack if a user inquires about who is using each of the computers on a LAN and then subsequently logs into one of the hosts. Detecting this type of behavior requires attributing multiple sessions, perhaps with different account names, to a single source.

This problem is unique to the network environment and has not been dealt with before in this context. Our solution to the multiple user identity problem is to create a *network-user identification* (NID) the first time a user enters the monitored environment, and then to apply that NID to any further instances of the user. All evidence about the behavior of any instance of the user is then accountable to the single NID. In particular, we must be able to determine that "smith@host1" is the same user as "jones@host2", if in fact they are. Since the network-user identification problem involves the collection and evaluation of data from both the host and LAN monitors, examining it is a useful method to understand the operation of DIDS. In the following subsections we examine each of the components of DIDS in the context of the creation and use of the NID.

5. The Host Monitor

The host monitor is currently installed on Sun SPARCstations running SunOS 4.0.x with the Sun C2 security package [9]. Through the C2 security package, the operating system produces audit records for virtually every transaction on the system. These transactions include file accesses, system calls, process executions, and logins. The contents of the Sun C2 audit record are: record type, record event, time, real user ID, audit user ID, effective user ID, real group ID, process ID, error code, return value, and label.

The host monitor (Fig. 3) examines each audit record to determine if it should be forwarded to the expert system for further evaluation. Certain critical audit records are always passed directly to the expert system (i.e., *notable events*); others are processed locally by the host monitor (i.e., *profiles* and attack *signatures*, which are sequences of noteworthy events which indicate the symptoms of attacks) and only summary reports are sent to the expert system. Thus, one of the design objectives is to push as much of the processing operations down to the low-level monitors as possible. In order to do this, the HEG creates a more abstract object called an *event*. The event includes any significant data provided by the original audit record plus two new fields: the *action* and the *domain*. The action and domain are abstractions which are used to minimize operating system dependencies at higher levels. Actions characterize the dynamic aspect of the audit records. Domains characterize the objects of the audit records. In most cases, the objects are files or devices and their domain is determined by the characteristics of the object or its location in the file system. Since processes can also be objects of an audit record, they are also assigned to domains, in this case by their function.

The actions are: session_start, session_end, read (a file or device), write (a file or device), execute (a process), terminate (a process), create (a file or (virtual) device), delete (a file or (virtual) device), move (rename a file or device), change_rights, and change_user_id. The domains are: tagged, authentication, audit, network, system, sys_info, user_info, utility, owned, and not_owned.

The domains are prioritized so that an object is assigned to the first applicable domain. *Tagged* objects are ones which are thought a priori to be particularly interesting in terms of detecting intrusions. Any file, device, or process can be tagged (e.g., */etc/passwd*). *Authentication* objects are the processes and files which are used to provide access control on the system (e.g., the password file). Similarly, *audit* objects relate to the accounting and security auditing processes and files. *Network* objects are the processes and files not covered in the previous domains which relate to the use of the network. *System* objects are primarily those which are concerned with the execution of the operating system itself, again exclusive of those objects already assigned to previously considered domains. *Sys_info* and *user_info* objects provide information about the system and about the users of the system, respectively. The *utility* objects are the bulk of the programs run by the users (e.g., compilers and editors). In general, the execution of an object in the utility domain is not interesting (except when the use is excessive), but the creation or modification of one is. *Owned* objects are relative to the user. *Not_owned* objects are, by exclusion, every object not assigned to a previous domain. They are also relative to a user; thus, files in the owned domain relative to "smith" are in the not_owned domain relative to "jones".

All possible transactions fall into one of a finite number of events formed by the cross product of the actions and the domains, and each event may also succeed or fail. Note that no distinction is made between files, directories or devices, and that all of these are treated simply as objects. Not every action is applicable to every object; for example, the *terminate* action is applicable only to processes. The choice of these domains and actions is somewhat arbitrary in that one could easily suggest both finer and coarser grained partitions. However, they capture most of the interesting behavior for intrusion detection and correspond reasonably well with what other researchers in this field have found to be of interest [5, 10]. By mapping an infinite number of transactions to a finite number of events, we not only remove operating system dependencies, but also restrict the number of permutations that the expert system will have to deal with. The concept of the domain is one of the keys to detecting abuses. Using the domain allows us to make assertions about the nature of a user's behavior in a straightforward and systematic way. Although we lose some details provided by the raw audit information, that is more than made up for by the increase in portability, speed, simplicity, and generality.

An event reported by a host monitor is called a host audit record (har). The record syntax is: har(Monitor-ID, Host-ID, Audit-UID, Real-UID, Effective-UID, Time, Domain, Action, Transaction, Object, Parent Process, PID, Return Value, Error Code).

Of all the possible events, only a subset are forwarded to the expert system. For the creation and application of the NID, it is the events which relate to the creation of user sessions or to a change in an account that are important. These include all the events with *session_start* actions, as well as ones with an *execute* action applied to the *network* domain. These latter events capture such transactions as executing the *rlogin, telnet, rsh,* and *rexec* UNIX programs. The HEG consults external tables, which are built by hand, to determine which events should be forwarded to the expert system. Because they relate to events rather than to the audit records themselves, the tables and the modules of the HEG which use them are portable across operating systems. The only portion of the HEG which is operating system dependent is the module which creates the events.

6. The LAN Monitor

The LAN monitor is currently a subset of UC Davis' Network Security Monitor [3]. The LAN monitor builds its own "LAN audit trail". The LAN monitor observes each and every packet on its segment of the LAN and, from these packets, it is able to construct higher-level objects such as connections (logical circuits), and service requests using the TCP/IP or UDP/IP protocols. In particular, it audits host-to-host connections, services used, and volume of traffic per connection.

Similar to the host monitor, the LAN monitor uses several simple analysis techniques to identify significant events. The events include the use of certain services (e.g., *rlogin* and *telnet*) as well as activity by certain classes of hosts (e.g., a PC without a host monitor). The LAN monitor also uses and maintains profiles of expected network behavior. The profiles consist of expected data paths (e.g., which systems are expected to establish communication paths to which other systems, and by which service) and service profiles (e.g., what a typical *telnet*, *mail*, or *finger* is expected to look like).

The LAN monitor also uses heuristics in an attempt to identify the likelihood that a particular connection represents intrusive behavior. These heuristics consider the capabilities of each of the network services, the level of authentication required for each of the services, the security level for each machine on the network, and signatures of past attacks. The abnormality of a connection is based on the probability of that particular connection occurring and the behavior of the connection itself. Upon request, the LAN monitor is also able to provide a more detailed examination of any connection, including capturing every character crossing the network (i.e., a wire-tap). This capability can be used to support a directed investigation of a particular subject or object. Like the host monitor, the LAN monitor forwards relevant security information to the director through its LAN agent.

An event reported by a LAN monitor is called a network audit record (nar). The record syntax is: nar(Monitor-ID, Source_Host, Dest_Host, Time, Service, Domain, Status).

The LAN monitor has several responsibilities with respect to the creation and use of the NID. The LAN monitor is responsible for detecting any connections related to *rlogin* and *telnet* sessions. Once these connections are detected, the LAN monitor can be used to verify the owner of a connection. The LAN monitor can also be used to help track tagged objects moving across the network. The SSO can also ask for a wire-tap on a certain network connection to monitor a particular user's behavior.

7. The Expert System

DIDS utilizes a rule-based (or production) expert system. The expert system is currently written in Prolog, and much of the form of the rule base comes from Prolog and the logic notation that Prolog implies. The expert system uses rules derived from the hierarchical Intrusion Detection Model (IDM). The IDM describes the data abstractions used in inferring an attack on a network of computers. That is, it describes the transformation from the distributed raw audit data to high level hypotheses about intrusions and about the overall security of the monitored environment. In abstracting and correlating data from the distributed sources, the model builds a virtual machine which consists of all the connected hosts as well as the network itself. This unified view of the distributed system simplifies the recognition of intrusive behavior which spans individual hosts. The model is also applicable to the trivial network of a single computer.

The model is the basis of the rule base. It serves both as a description of the function of the rule base, and as a touchstone for the actual development of the rules. The IDM consists of 6 layers, each layer representing the result of a transformation performed on the data (see Table 1).

The objects at the first level of the model are the audit records provided by the host operating system, by the LAN monitor, or by a third party auditing package. The objects at this level are both syntactically and semantically dependent on the source. At this level, all of the activity on the host or LAN is represented.

At the second level, the *event* (which has already been discussed in the context of the host and LAN monitor) is both syntactically and semantically independent of the source standard format for events.

The third layer of the IDM creates a *subject*. This introduces a single identification for a user across many hosts on the network. It is the subject who is identified by the NID (see section 7.1). Upper layers of the model treat the network-user as a single entity, essentially ignoring the local identification on each host. Similarly, above this level, the collection of hosts on the LAN are generally treated as a single distributed system with little attention being paid to the individual hosts.

The fourth layer of the model introduces the event in *context*. There are two kinds of context: temporal and spatial. As an example of temporal context, behavior which is unremarkable during standard working hours may be highly suspicious during off hours [5]. The IDM, therefore, allows for the application of information about wall-clock time to the events it is considering. Wall-clock time refers to information about the time of day, weekdays versus weekends and holidays, as well as periods when an increase in activity is expected. In addition to the consideration of external temporal context, the expert system uses time windows to correlate events occurring in temporal proximity. This notion of temporal proximity implements the heuristic that a call to the UNIX *who* command followed closely by a *login* or *logout* is more likely to be related to an intrusion than either of those events occurring alone. Spatial context implies the relative importance of the source of events. That is, events related to a particular user, or events from a particular host, may be more likely to represent an intrusion than similar events from a different source. For instance, a user moving from a low-security machine to a high-security machine may be of greater concern than a user moving in the opposite direction. The model also allows for the correlation of multiple events from the same user or source. In both of these cases, multiple events are more noteworthy when they have a common element than when they do not.

The fifth layer of the model considers the *threats* to the network and the hosts connected to it. Events in context are combined to create threats. The threats are partitioned by the nature of the abuse and the nature of the target. In other words, what is the intruder doing, and what is he doing it to? Abuses are divided into *attacks*, *misuses*, and *suspicious acts*. Attacks represent abuses in which the state of the machine is changed. That is, the file system or process state is different after the attack than it was prior to the attack. Misuses represent out-of-policy behavior in which the state of the machine is not affected. Suspicious acts are events which, while not a violation of policy, are of interest to an IDS. For example, commands which provide information about the state of the system may be suspicious. The targets of abuse are characterized as being either *system* objects or *user* objects and as being either *passive* or *active*. User objects are owned by non-privileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including executable binaries, while active objects are essentially running processes.

At the highest level, the model produces a numeric value between one and 100 which represents the overall *security state* of the network. The higher the number the less secure the network. This value is a function of all the threats for all the subjects on the system. Here again we treat the collection of hosts as a single distributed system. Although representing the security level of the system as a single value seems to imply some loss of information, it provides a quick reference point for the SSO. In fact, in the current implementation, no information is lost since the expert system maintains all the evidence used in calculating the security state in its internal database, and the SSO has access to that database.

In the context of the network-user identification problem we are concerned primarily with the lowest three levels of the model: the audit data, the event, and the subject. The generation of the first two of these have already been discussed; thus, the creation of the subject is the focus of the following subsection.

The expert system is responsible for applying the rules to the evidence provided by the monitors. In general, the rules do not change during the execution of the expert system. What does change is a numerical value associated with each rule. This *Rule Value* (RV) represents our confidence that the rule is useful in detecting intrusions. These rule values are manipulated using a negative reinforcement training method which allows the expert system to continually lower the number of false attack reports. When a potential attack is reported by the expert system, the SSO determines the validity of the report and gives feedback to the expert system. If the report was deemed faulty, then the expert system lowers the RV's associated with the rules that were used to draw that conclusion. In addition to this directed training, which may lower some rule values, the system also automatically increases the RV's of all the rules on a regular basis. This recovery algorithm allows the system to adapt to changes in the environment as well as recover from faulty training.

Logically the rules have the form:

antecedent => consequence

where the antecedent is either a fact reported by one of the distributed monitors, or a consequence of some previously satisfied rule. The antecedent may also be a conjunction of these. The overall structure of the rule base is a tree rooted at the top. Thus, many facts at the bottom of the tree will lead to a few conclusions at the top of the tree.

The expert system shell consists of approximately a hundred lines of Prolog source code. The shell is responsible for reading new facts reported by the distributed monitors, attempting to apply the rules to the facts and hypotheses in the Prolog database, reporting suspected intrusions, and maintaining the various dynamic values associated with the rules and hypotheses. The syntax for rules is:

rule(n,r,(single,[A]),(C))).

where n is the rule number, r is the initial RV, A is the single antecedent, and C is the consequence. Conjunctive rules have the form:

rule(n,r,(and,[A_1,A_2,A_3]),(C))).

where A_1,A_2,A_3 are the antecedents and C is the consequence. Disjunctive rules are not allowed; that situation is dealt with by having multiple rules with the same consequence.

7.1. Building the NID

With respect to Unix, the only legitimate ways to create an instance of a user are for the user to login from a terminal, console, or off-LAN source, to change the user-id in an existing instance, or to create additional instances (local or remote) from an existing instance. In each case, there is only one initial login (system wide) from an external device. When this original login is detected, a new unique NID is created. This NID is applied to every subsequent action generated by that user. When a user with a NID creates a new login session, that new session is associated with his original NID. Thus the system maintains a single identification for each physical user.

We consider an instance of a user to be the 4-tuple *<session_start, user-id, host-id, time>*. Thus each login creates a new instance of a user. In associating a NID with an instance of a user, the expert system first tries to use an existing NID. If no NID can be found which applies to the instance, a new one is created. Trying to find an applicable existing NID consists of several steps. If a user changes identity (e.g., using UNIX's *su* command) on a host, the new instance is assigned the same NID as the previous identity. If a user performs a remote login from one host to another host, the new instance gets the same NID as the source instance. When no applicable NID is found, a new unique NID is created by the following rule:

```
rule(111,1000,[
   hhar(_,Host1,AUID,_,_,Time1,_,session_start,_,_,'local',_,_,_),  /* login */
   \+ (ih(net_user(NID,AUID,Host,_),_,_,_)),  /* no NID yet */
   newNID(X)  /* create new NID */
],
   (net_user(X,AUID,Host1,Time1))).  /* new net user */
```

The actual association of a NID with a user instance is through the hypothesis *net_user*. A new hypothesis is created for every event reported by the distributed monitors. This new hypothesis, called a *subject*, is formed by the rule:

```
rule(110,100,(and,[
  har(Mon,Host,AUID,UID,EUID,Time,Dom,Act,Trans,Obj,Parent,PID,Ret,Err).
  net_user(NID,AUID,Host,_)
]),
  subj(NID,Mon,Host,AUID,UID,EUID,Time,Dom,Act,Trans,Obj,Parent,PID,Ret,Err))).
```

The rule creates a subject, getting the NID from the net_user and the remaining fields from the host audit record, if and only if both the user-id and the host-id match. It is through the use of the subject that the expert system correlates a user's actions regardless of the login name or host-id.

There is still some uncertainty involved with the network-user identification problem. If a user leaves the monitored domain and then comes back in with a different user-id, it is not possible to connect the two instances. Similarly, if a user passes through an unmonitored host, there is still uncertainty that any connection leaving the host is attributable to any connection entering the host. Multiple connections originating from the same host at approximately the same time also allow uncertainty if the user names do not provide any helpful information. The expert system can make a final decision with additional information from the host and LAN monitors that can (with high probability) disambiguate the connections.

8. Conclusion

Our Distributed Intrusion Detection System (DIDS) is being developed to address the shortcomings of current single host IDS's by generalizing the target environment to multiple hosts connected via a network (LAN). Most current IDS's do not consider the impact of the LAN structure when attempting to monitor user behavior for attacks against the system. Intrusion detection systems designed for a network environment will become increasingly important as the number and size of LAN's increase. Our prototype has demonstrated the viability of our distributed architecture in solving the network-user identification problem. We have tested the system on a sub-network of Sun SPARCstations and it has correctly identified network users in a variety of scenarios. Work continues on the design, development, and refinement of rules, particularly those which can take advantage of knowledge about particular kinds of attacks. The initial prototype expert system has been written in Prolog, but it is currently being ported to CLIPS due to the latter's superior performance characteristics and easy integration with the C programming language. We are designing a signature analysis component for the host monitor to detect events and sequences of events that are known to be indicative of an attack, based on a specific context. In addition to the current host monitor, which is designed to detect attacks on general purpose multi-user computers, we intend to develop monitors for application specific hosts such as file servers and gateways. In support of the ongoing development of DIDS we are planning to extend our model to a hierarchical Wide Area Network environment.

Acknowledgments

The DIDS project is sponsored by the United States Air Force Cryptologic Support Center through a contract with the Lawrence Livermore National Labs.

References

1. Department of Defense, *Trusted Computer System Evaluation Criteria,* National Computer Security Center, DOD 5200.28-STD, Dec. 1985.
2. G.V. Dias, K.N. Levitt, and B. Mukherjee, "Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection," Technical Report CSE-90-41, University of California, Davis, Jul. 1990.
3. L.T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A Network Security Monitor," *Proc. 1990 Symposium on Research in Security and Privacy,* pp. 296-304, Oakland, CA, May 1990.
4. B. Landreth, *Out of the Inner Circle, A Hacker's Guide to Compuer Security,* Microsoft Press, Bellevue, WA, 1985.
5. T. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," *Proc. 11th National Computer Security Conference,* pp. 65-73, Baltimore, MD, Oct. 1988.
6. T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P.G. Neumann, and C. Jalali, "IDES: A Progress Report," *Proc. Sixth Annual Computer Security Applications Conference,* Tucson, AZ, Dec. 1990.
7. T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H.S. Javitz, A. Valdes, and P.G. Neumann, "A Real-Time Intrusion-Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International, May 1990.
8. M.M. Sebring, E. Shellhouse, M.E. Hanna, and R.A. Whitehurst, "Expert Systems in Intrusion Detection: A Case Study," *Proc. 11th National Computer Security Conference,* pp. 74-81, Oct. 1988.
9. W.O. Sibert, "Auditing in a Distributed System: SunOS MLS Audit Trails," *Proc. 11th National Computer Security Conference,* Baltimore, MD, Oct. 1988.
10. S.E. Smaha, "Haystack: An Intrusion Detection System," *Proc. IEEE Fourth Aerospace Computer Security Applications Conference,* Orlando, FL, Dec. 1988.
11. H.S. Vaccaro and G.E. Liepins, "Detection of Anomalous Computer Session Activity," *Proc. 1989 Symposium on Research in Security and Privacy,* pp. 280-289, Oakland, CA, May 1989.
12. J.R. Winkler, "A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks," *Proc. 13th National Computer Security Conference,* pp. 115-124, Washington, D.C., Oct. 1990.

Level	Name	Explanation
6	Security State	overall network security level
5	Threat	definition of categories of abuse
4	Context	event placed in context
3	Subject	definition and disambiguation of network user
2	Event	OS independent representation of user action (finite number of these)
1	Data	audit or OS provided data

Table 1. Intrusion Detection Model

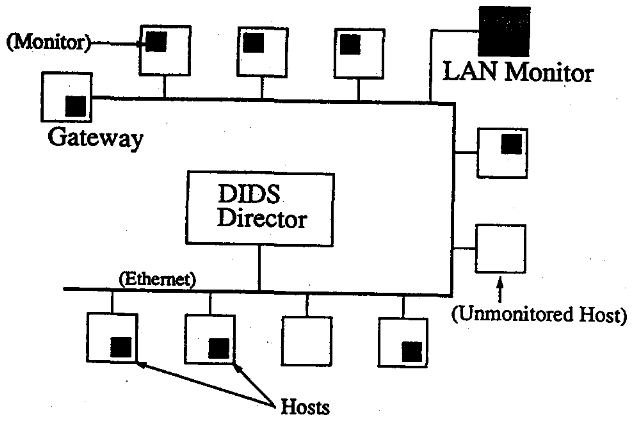


Fig. 1. DIDS Target Environment

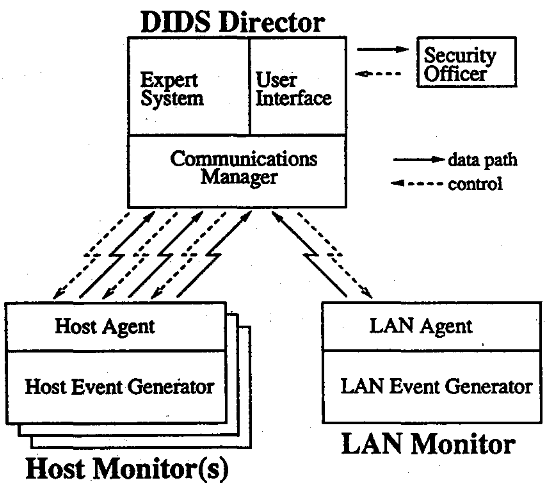


Fig. 2. Communications Architecture

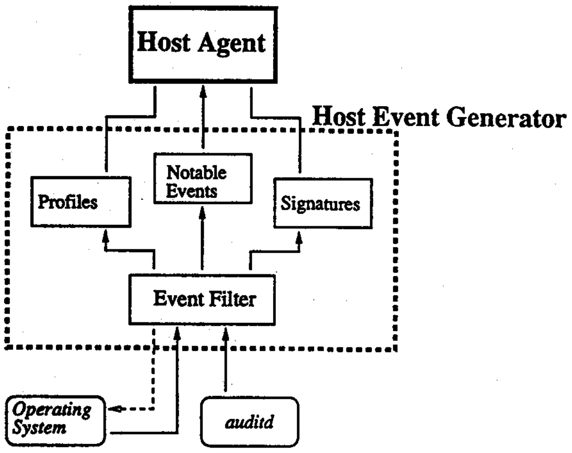


Fig. 3. Host Monitor Structure

A DISTRIBUTED IMPLEMENTATION OF THE TRANSFORM MODEL

Ravi S. Sandhu and Gurpreet S. Suri

Center for Secure Information Systems
and
Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030-4444

ABSTRACT The Transform access-control model is based on the concept of transformation of access rights. It has previously been shown that Transform unifies a number of diverse access control mechanisms such as amplification, copy flags, separation of duties and synergistic authorization. It has also been shown that Transform has an efficient algorithm for safety analysis of the propagation of access rights (i.e., the determination of whether or not a given subject can ever acquire access to a given object). In this paper we propose a distributed implementation of Transform. Our design is based on capabilities with identities of subjects buried in them. This ensures unforgeability of capabilities as well as enables enforcement of "mandatory" controls on propagation of capabilities from one subject to another. The design provides for immediate, selective, partial and complete revocation on a temporary as well as permanent basis.

Keywords: Distributed Systems, Secure Architectures, Capabilities

1 INTRODUCTION

The need for access controls arises in any computer system that provides for controlled sharing of information and other resources among multiple users. Access control models (or protection models) provide a framework for specifying, analyzing and implementing security policies in multiuser systems. These models are typically defined in terms of the well-known abstractions of subjects, objects and access rights with which we assume the reader is familiar. A wide variety of access-control models have been described in the literature [3,4,10,12,16, for instance]. Unfortunately very few have been implemented or have even influenced implementations of actual systems.*

In this paper we take a step towards closing this gap between theory and practise. Our principal contribution is the outline of a distributed implementation of the recently proposed Transform model [17]. Transform derives its name from its central concept of transformation of access rights. The idea is that access rights get transformed as they are propagated from one subject to another, e.g., a security-officer who has the review right for a document may propagate the release right for the document to the document's author. It has previously been shown [17] that Transform elegantly unifies a number of seemingly different access control mechanisms such as amplification [5], copy flags [12], separation of duties [4] and synergistic authorization [14]. It has also been shown [17] that there are efficient algorithms for the safety problem in Transform (i.e., the determination of whether or not a given subject can ever acquire access to a given object).

Thus Transform incorporates practically useful expressive power while allowing for safety analysis. Transform is actually a special case of the Schematic Protection Model (SPM) [16]. Like Transform, SPM also exhibits strong safety properties. This is in contrast to the weak safety properties of the access-matrix model commonly known as HRU [10]. Both HRU and SPM have undecidable safety in general [10,18]. In HRU safety becomes undecidable under very weak assumptions, notably

*The notable exception is the Bell-LaPadula model [3] whose strong influence on military systems has been formally incorporated in evaluation criteria [8].

the bi-conditional monotonic case of [11]. On the other hand safety in SPM remains decidable under very strong assumptions, notably the acyclic attenuating case of [16]. In particular Transform falls outside the known decidable cases for HRU but well within the known decidable cases for SPM [17].

Our implementation proposal for Transform is strongly influenced by the identity-based capability architecture proposed by Gong [9]. The concept of embedding the identity of a subject in a capability in distributed systems has been known for some time [6]. It ensures that capabilities cannot be forged or propagated from one subject to another without intervention of trusted software. Gong's architecture is based on the familiar client-server model of services in a distributed system and includes mechanisms for revocation which were missing in earlier proposals such as [6]. We have extended Gong's proposal to accommodate Transform. In particular the concept of strongly typed subjects and objects, which is essential to Transform, has been incorporated.

The paper is organized as follows. Section 2 reviews the Transform model to the extent required for our objectives in this paper. Section 3 discusses distributed capability-based architectures in general and motivates our choice of building on Gong's approach. Section 4 describes our proposed implementation for Transform. The protocols involved in creation, propagation and revocation are presented. An example of the implementation is presented in section 5. The paper is concluded in section 6 with a discussion and proposals for future research.

2 THE TRANSFORM MODEL

The Transform model [17] was obtained by identifying the common foundation underlying a variety of different access-control mechanisms proposed in the literature. These include amplification [5], copy flags [12], separation of duties [4] and synergistic authorization [14]. Considered in isolation these mechanisms are diverse and were largely proposed independently of each other. They all appear to be desirable and should be supported by any system which claims generality. However simply lumping them together results in a complex system with many unrelated mechanisms.

Transform introduces the unifying concept of transformation of rights which can occur in two different ways.

1. *Self transformation* or *internal transformation* allows a subject who possesses certain rights for an object to obtain additional rights for that object.

2. *Grant transformation* or *external transformation* occurs in the granting of access rights by one subject to another. The general idea is that possession of a right for an object by a subject allows that subject to give some other right for that object to another subject.

In addition Transform is based on the *strong typing* of subjects and objects, i.e., subjects and objects are classified into types when they are created and their type cannot change thereafter. Much of the power of transformation derives from predicating the ability to transform on the types of subjects and objects involved.

A security policy is stated in Transform by specifying the following (finite) components.

1. Disjoint sets of subject types TS object types TO and rights R.

2. A can-create function $cc : \mathrm{TS} \rightarrow 2^{\mathrm{TO}}$.

3. Create-rules $cr : \mathrm{TS} \times \mathrm{TO} \rightarrow 2^{\mathrm{R}}$.

4. An internal transformation function $itrans : \mathrm{TS} \times \mathrm{TO} \times 2^{\mathrm{R}} \rightarrow 2^{\mathrm{R}}$.

5. A grant transformation function $grant : \mathrm{TS} \times \mathrm{TS} \times \mathrm{TO} \times 2^{\mathrm{R}} \rightarrow 2^{\mathrm{R}}$.

The notation 2^X denotes the power set of X, i.e., the set of all subsets of X. These components of a Transform specification are explained in turn below.

The sets TS and TO define the subject types and object types respectively. For example subject types might be faculty, student, guest, etc., and object types might be file, mail-message, bulletin-board, etc. R defines the set of rights or privileges in the system, e.g., read, write, execute, etc.

There are two issues involved in object creation.[†] Firstly subjects need authorization to create objects. Secondly the rights obtained as a result of creation must also be specified. Transform authorizes creation by means of the can-create function *cc*. The interpretation of

$$cc(\mathrm{u}) = \{\mathrm{o}_1, \mathrm{o}_2, \ldots, \mathrm{o}_k\}$$

is that a subject of type u is authorized to create objects of type o_1 and objects of type o_2, etc. The effect of creation is defined by create-rules. The interpretation of

$$cr(\mathrm{u}, \mathrm{o}) = \{\mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_p\}$$

is that when a subject U of type u creates an object O of type o the creator U obtains the rights r_1, r_2, ..., r_p for O. For example if *cc*(user) = {file} and *cr*(user,file) = {own} the creator of a file gets the own right for it. For readability we will usually drop the set parenthesis around singleton sets, for instance by writing *cc*(user) = file and *cr*(user,file) = own.

Authorization for internal transformation is specified by the internal transformation function *itrans*. The interpretation of

$$itrans(\mathrm{u}, \mathrm{o}, \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}) = \{\mathrm{y}_1, \ldots, \mathrm{y}_m\}$$

is that a subject of type u who has *all* the x_i rights specified on the left hand side for an object of type o can obtain the rights y_1, ..., y_m for that object by internal transformation. For example, the policy that possession of the w (write) privilege for a file implies possession of the r (read) privilege is easily stated as follows.[‡]

$$itrans(\mathrm{user}, \mathrm{file}, \mathrm{w}) = \mathrm{r}$$

Another example of internal transformation occurs in situations described as synergistic authorization in [14]. For instance consider a situation where a scientist (abbreviated as sci) needs approvals from a security officer and a patent officer before he can release a document (abbreviated as doc) for publication. Say these two approvals are respectively signified by possession of the a_s and a_p rights. We can express this policy as follows.

$$itrans(\mathrm{sci}, \mathrm{doc}, \{\mathrm{own}, \mathrm{a}_s, \mathrm{a}_p\}) = \mathrm{release}$$

That is, a scientist who owns a document and possesses the two approvals can acquire the release right for that document.

Grant transformations are authorized by the grant transformation function *grant*. The interpretation of

$$grant(\mathrm{u}, \mathrm{v}, \mathrm{o}, \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}) = \{\mathrm{y}_1, \ldots, \mathrm{y}_m\}$$

[†]There must be provision for creation of subjects in any realistic system. In practise creation of subjects is often strictly controlled by some distinguished system administrator or security officer. Such creation can be considered as occurring outside the normal scope of the system.

[‡]In multilevel systems this policy would amount to prohibiting write-up.

is that a subject of type u who has *all* the x_i rights specified on the left hand side for an object of type o can grant *one or more* of the rights $y_1, \ldots, y_m$ for that object to a subject of type v. A common example of grant transformation occurs with the copy flag c which controls whether the granted privilege can itself be further granted or not. For instance the following

$$\begin{aligned} \textit{grant}(\text{user, user, file, xc}) &= \{\text{xc, x}\} \\ \textit{grant}(\text{user, user, file, x}) &= \phi \end{aligned}$$

defines the (unlimited) copy flag. Here a user who has the xc privilege for a file can grant the xc privilege or the x privilege to another user, whereas a user with the x privilege for the file cannot grant x any further. Other variations of the copy flag, such as 1-step or n-step copy flags can be similarly defined [17].

The expressive power of Transform is illustrated by the following policy specification.

$$\begin{aligned} \textit{cc}(\text{sci}) &= \text{doc} \\ \textit{cr}(\text{sci, doc}) &= \{\text{own, read}\} \\ \textit{grant}(\text{sci, security-officer, doc, own}) &= \text{review} \\ \textit{grant}(\text{sci, patent-officer, doc, own}) &= \text{review} \\ \textit{grant}(\text{security-officer, sci, doc, review}) &= a_s \\ \textit{grant}(\text{patent-officer, sci, doc, review}) &= a_p \\ \textit{itrans}(\text{sci, doc}, \{\text{own}, a_s, a_p\}) &= \text{release} \end{aligned}$$

The first two equations specify that (i) a scientist can create documents, and (ii) the scientist who creates a document obtains the own and read privileges for it.[§] The next two equations specify that a scientist who owns a document can ask for it to be reviewed by a security-officer and by a patent-officer. These officers can respectively return the a_s and a_p rights to the scientist signifying the respective approvals. The scientist can then release the document. This example is further elaborated in section 5.

This completes our description of the Transform model. Further motivation for Transform and additional examples of policies are given in [17].

3 DISTRIBUTED CAPABILITY SYSTEMS

Capability-based architectures have had a strong appeal ever since the concept was first proposed [7]. They are viewed as providing a sound and common basis for providing both reliability and security. In the context of conventional centralized systems a number of such machines have been built [13]. Some have even achieved moderate commercial success. Nevertheless today's popular CPUs are not capability based. In retrospect one can argue that using capabilities to solve the memory protection problem is an overkill. The marginal advantages of capabilities over memory segmentation and protection rings (which are available in the latest generation of microprocessors such as the Intel 80386) do not justify the extra costs and performance penalties. In other words the initial application of capabilities was at too low a level.

It is expected by many researchers [15, for instance] that in the 1990s distributed operating systems will dominate the computing environment. These systems will appear to users as a single centralized system with complete location transparency. To achieve this, reliability and security must be addressed as part of the basic design of these systems. Attempts to graft security features

[§]Once a document has been created it can no longer be written. This is necessary in order to freeze the contents of the document. If revisions are required a new version of the document needs to be created.

later in the design cycle will surely fail, much as they are failing in conventional centralized systems. The capability-based framework continues to offer an attractive approach to these problems. In a distributed operating system capabilities are introduced at a much higher level than memory addressing. Capabilities need to be incorporated into the remote procedure call mechanism rather than the memory addressing mechanism. This offers the hope that the additional overhead will not severely degrade performance. Capabilities can moreover be integrated into the basic client-server structure of distributed systems to provide transparency.

There are three basic issues which must be confronted by the designer of a distributed capability-based system. These issues are complicated relative to conventional centralized capability-based systems because capabilities are dispersed in individual workstations and can no longer be assumed to be under tight control of a security kernel.

1. *Unforgeability.* It must be guaranteed that capabilities cannot be modified or manufactured by subjects. This requires some form of cryptographic sealing.

2. *Propagation.* It must be guaranteed that capabilities cannot be copied from one user to another. This requires some means of embedding the identity of a subject in a capability.

3. *Revocation.* It must be guaranteed that capabilities which have been granted can be withdrawn or revoked in a timely manner. This requires some means of invalidating existing capabilities and accounting for cascaded revocation.

Various solutions to one or more of these problems have been proposed in the literature. For instance Amoeba [15] uses "sparse capabilities" with cryptographic protection to ensure unforgeability. Unfortunately Amoeba does not address capability propagation or revocation. Davies [6] discusses mechanisms to embed the identity of a subject in a capability. This ensures that capabilities cannot be forged or propagated from one subject to another without intervention of trusted software. Davies, however, does not address the revocation issue. Gong's proposed architecture [9] is the first attempt to address all three issues in a distributed context. It is based on the familiar client-server model of services in distributed systems and therefore is a suitable foundation for us to build upon. However, Gong does not incorporate the notion of types which is basic to Transform. His architecture therefore needs to be extended for this purpose.

4 IMPLEMENTATION OF TRANSFORM

We now describe a distributed capability-based implementation of the Transform model. We assume that objects are encapsulated within object servers. The basic computation model is that of remote procedure calls involving the following sequence of events: (i) a client sends a request to a server to manipulate one or more objects, (ii) the server accepts and services the request, and (iii) the server sends back a reply. The object server runs on a trusted host which guarantees that the server cannot be bypassed. For ease of exposition we visualize each object server as running on a separate host. However, we allow multiple object servers on the same trusted host provided the security kernel on the host can enforce separation among these servers. If we have sufficient confidence in the security kernel we can also allow untrusted clients to coexist with object servers on a single trusted host.

Each object server acts as the reference monitor (or access mediator) for the set of objects it manages. In other words the object server is part of the trusted computing base (TCB). The object server is responsible not only for access mediation but also for ensuring semantic correctness of the objects with respect to the abstract operations exported from the server. The object server itself has the ability to access all objects within its control. We emphasize that the object server is not a subject in the system but is rather a part of the TCB.

For simplicity, we require that each object server manage exactly one type of object. In practise this rule would probably be relaxed to allow a single server to manage multiple object types, particularly if they are closely related. On the other hand the same type of object may be managed by multiple object servers. For instance a given system may have numerous file servers. An individual file server manages some subset of the total collection of files in the system. We assume there is no replication of files, i.e., each file resides at exactly one file server.

Finally we assume there is an access decision facility (ADF) which can be consulted by object servers to determine the security policy. In the context of Transform the ADF will be consulted by object servers for finding out appropriate values of *cc*, *cr*, *grant* and *itrans*. Pieces of the ADF may actually reside at each object server while other pieces are remotely accessed. The reason for this is to allow quick local access to well-established and relatively static aspects of the policy while at the same time allowing for new types etc. to be introduced.

4.1 Identity and Type

Each subject or object in the system has a globally unique identifier. Each subject or object also has a unique type which is determined when that subject or object is created. Thereafter the type cannot change. We assume the type of a subject or object is embedded in its identifier. Henceforth we refer to a subject identifier by *sid* and a object identifier by *oid*. These identifiers have the following structure.

type	identifier

The type field denotes the type of the object while the identifier field uniquely identifies each subject or object among instances of the same type. Note that sid's and oid's can be generated at will by users.

4.2 Capability Seeds

A capability seed is a secret random number associated with each oid. The seed is known only to the object server which manages the object identified by oid. We can visualize this association by the following pair.¶

oid	seed

The purpose of the seed is to facilitate revocation and prevent against replay of revoked capabilities, as will be discussed later.

4.3 Capabilities

A capability has the following structure.

oid	rights	seal

where the seal is computed using a publicly known one-way function f as follows.

$$\text{seal} = f(\text{sid}, \text{oid}, \text{rights}, \text{seed})$$

¶Gong [9] calls this pair an "internal capability." We feel the name "internal capability" is a misnomer and prefer to call the secret random number a capability seed because its principal use is in cryptographically sealing capabilities exported from the object server.

The oid and rights components of a capability are exactly as one would expect even in a conventional centralized system. The seal cryptographically embeds the subject identifier (sid) in the capability using the capability seed for that purpose.

4.4 Access Mediation

Access mediation must be incorporated into the RPC (Remote Procedure Call) mechanism of the client-server architecture. The object server must authenticate the source of every RPC request. For this purpose, we assume that each subject has the means to place its digital signature on every RPC communication to a object server. The RPC also carries within it the relevant capabilities for the operation being requested. The object server first verifies that the sid on each capability is authenticated by the digital signature, otherwise the RPC is immediately rejected. Then the object server looks up the capability seed for oid, computes the seal using the above formula and compares the computed seal with the seal submitted by the subject. If these match the capability is known to be authentic and the operation is performed provided the rights are sufficient to authorize it. Digital signatures for the reverse communication from object servers to subjects can also be incorporated. The details of these protocols are beyond the scope of this paper and can readily be found in the standard literature [1, for instance]. We envisage a implementation similar to the interface function box of Amoeba [15] which are placed between each processor module and the network.

4.5 Creation

For object creation the object server consults the access decision facility (ADF) to determine whether or not such creation is authorized by *cc*(sid.type). If the creation is authorized a new object is created with a new oid and a new capability seed. The rights to be entered on the capability are determined from *cr*(sid.type,oid.type). Finally the capability is sealed and returned to the subject.

4.6 Internal Transformation

Let subject sid request the following internal transformation for object oid.

$$\textit{itrans}(\mathrm{u}, \mathrm{o}, \{\mathrm{x}_1,\ldots,\mathrm{x}_n\}) = \{\mathrm{y}_1,\ldots,\mathrm{y}_m\}$$

The object server must, of course, be a manager for objects of type o. The server checks that sid.type=u and oid.type=o. It also checks that the RPC request includes a capability (or capability list) for object oid with the rights $x_1, \ldots, x_n$. This check is performed by comparing the computed seal with the seal on the capability as discussed in section 4.4. Finally the object server creates a new capability sealed for sid with rights $x_1, \ldots, x_n$ $y_1, \ldots, y_m$. This capability is returned to the subject sid. Note that the original capability, with rights $x_1, \ldots, x_n$ continues to be valid. It is however redundant and can be discarded by the subject.

4.7 Grant Transformation

Let subject sid1 request the following grant transformation for object oid to subject sid2.

$$\textit{grant}(\mathrm{u}, \mathrm{v}, \mathrm{o}, \{\mathrm{x}_1,\ldots,\mathrm{x}_n\}) = \{\mathrm{y}_1,\ldots,\mathrm{y}_m\}$$

The object server should again be a manager for objects of type o. The server checks that sid1.type=u, sid2.type=v and oid.type=o. It also checks that the RPC request includes a capability (or capability list) for object oid with the rights $x_1, \ldots, x_n$. If the check is successful the object server creates a new capability sealed for sid2 with rights $y_1, \ldots, y_m$. This capability is returned to the subject sid1 who can then pass it on to subject sid2.

4.8 Revocation

Revocation has always been a problem in capability-based systems. In distributed systems the problem is further compounded, since the subjects are completely autonomous with no centralized authorities enforcing security. There are various issues against which the implementation of revocation can be compared [19].

1. Partial or Complete: Whether it is possible to revoke a specific right or whether all rights in a capability have to be revoked to get any sort of denial of access in the system?
2. Immediate or Delayed: If the implementation executes revocation immediately or it comes into force only the next time the subject tries to access the object?
3. Selective or General: Does the revocation process affect all users or a select group of users having access over the object?
4. Temporary or Permanent: Is access is to be denied permanently or if once it is revoked, is it retrievable?

We provide revocation by a revocation list and a count field appended to the seed as shown below.

oid	seed	count	revocation list

The revocation list contains entries of sids for whom the rights for that particular oid have been revoked. The list specifies for each sid which of its rights have been revoked. When the validity of the capability is checked during access mediation, the revocation lists are checked in parallel as well. Since access mediation is performed on every operation revocation is immediate. The owner of an oid always has the option to revoke partially or completely the capability of a sid for that oid. Partial or complete revocation of a sid in no way interferes with the access rights of other sids.

The count is a measure that determines the number of valid capabilities for that seed. The count is incremented during creation and propagation, but decremented during complete revocation (i.e. when all the rights of a subject for that object are revoked). Temporary or permanent revocation is carried out, depending on the value of the count. If the size of the revocation list becomes a significant fraction of the count the object server goes ahead with permanent revocation. The server deletes the seed associated with that oid, computes a new one and sends new recomputed capabilities to other associated sids. This of course requires that the object server keep a log of propagation of capabilities. However if the size of the revocation list is small in comparison to the count, the object server goes ahead with temporary revocation. In this case the object server appends the revocation information onto the revocation list associated with that oid.

5 EXAMPLE

The scientist and the security-officer example discussed earlier in section 2 is illustrated here using the protocols described above. A scientist (say Joe) creates a document (say SDI) on his workstation, but before he can release it he needs to have approval from a security-officer (say Sam) and a patent-officer (say Pat). The following is the sequence of protocols needed to complete the task.

1. Joe asks the server to create a document called SDI. This RPC is made by the kernel of Joe's workstation to the appropriate daemon responsible for the server's actions. The RPC contains the action requested, the sid, oid, the types of sid and oid involved, and the actual data to be stored in the created document; all signed under Joe's digital signature. In this case the sid=sci.Joe and the oid=doc.SDI. Joe and SDI are respectively of type sci and doc. On receiving the request, server checks the digital signature to authenticate Joe. The server then checks the *cc* policy, taking into account the sid, oid and their types provided. If it is in the affirmative it checks the *cr* policy, by which it determines what rights Joe gets for the document he is creating. The server then pulls out the seed say seed1 for that document and stores it in its internal tables with the following association:

doc.SDI	seed1

   Then the object server manufactures the following capability and sends it to Joe (strictly speaking to the kernel of Joe's workstation):

doc.SDI	own, read	seal1

   where seal1 = f(sci.Joe, doc.SDI, {own, read}, seed1)

2. Now Joe is ready to release the document. His workstation sends the propagation requests to the server on his behalf. The RPC looks like this:

grant(Sam, review)	doc.SDI	own, read	seal1

   The host when framing the RPC, appends to it the capability it possesses for SDI and signs the request under Joe's digital signature. The server on receiving the request verifies the digital signature and authenticates Joe. Then the server checks the validity of the capability by retrieving the seed of SDI, i.e. seed1, from its internal tables, and computing the seal using the one way function f. Then it computes seal1 from the capability provided by Joe and if the two seals match the validity of the capability is confirmed. The request is then checked against the *grant* policy of Transform. When the server determines Joe has sufficient rights, i.e. own, for SDI, it authorizes the grant. The server then computes the capability for the security-officer Sam to have the review right for SDI. The capability

doc.SDI	review	seal2

   where seal2 = f(security-officer.Sam, doc.SDI, review, seed1)

   is sent to Joe. Joe then forwards this capability to Sam. Sam now has the capability for oid=doc.SDI with the review right. With this capability he can only access the document to review it. If he tries to get additional rights by internal transformation, the server will turn down his request because when it will check the set of rights he possesses, namely review, which is insufficient set for it to grant him additional rights. Sam now reviews the document, and if he approves of the action to release SDI he requests the server to grant Joe the approval (a_s) right.

grant(sci.Joe, a_s)	doc.SDI	review	seal2

   The server computes the following capability and sends it back to Sam who in turn sends it to Joe.

doc.SDI	a_s	seal3

where seal3 = f(sci.Joe, doc.SDI, a_s, seed1)

3. Exact similar protocol steps are executed to get the approval (a_p) from the patent-officer Pat. At the end of this session Joe possesses the following capability.

doc.SDI	a_p	seal4

where seal4 = f(sci.Joe, doc.SDI, a_p, seed1)

4. Now the scientist Joe possesses the capabilities giving him the approval to get the release right by internal transformation. Joe presents these capabilities to the server with the following request:

	doc.SDI	own, read	seal1
itrans(release)	doc.SDI	a_s	seal3
	doc.SDI	a_p	seal4

Like before, the server carries out the authentication and the validity tests on the capabilities presented to it by Joe. Then the server checks that Joe has the rights own, a_s and a_p for SDI which are required to get the additional release right. The server sends him a new capability:

doc.SDI	own, read, a_s, a_p, release	seal5

where seal5 = f(sci.Joe, doc.SDI, {own, read, a_s, a_p, release}, seed1)

This completes the example.

6 CONCLUSION

In this paper we have proposed a distributed capability-based implementation for the Transform model. The system is based on object servers who act as access-mediators on any attempt by a subject to create, use, acquire, grant or revoke capabilities. We assume a digital signature facility which authenticates the originating subject on each remote procedure call. The capabilities are cryptographically sealed to tie together the identity of the subject, the identity of the object, the rights and a secret cryptographic seed. Strong typing of subjects and objects has also been incorporated.

Our long term goal is to arrive at a practical distributed implementation for SPM (and its recent extension called ESPM [2]). Our first step towards this goal is the implementation of Transform described here. Transform is a sufficiently interesting and non-trivial special case of SPM. At the same time Transform is a sufficiently simplified version of SPM for which a realistic near-term implementation can be contemplated.

Acknowledgment

We are indebted to Howard Stainer and Sylvan Pinsky for their support and encouragement, making this work possible. The opinions expressed in this paper are of course our own and should not be taken to represent the views of these individuals.

References

[1] Akl, S.G. "Digital Signatures: A Tutorial Survey." *Computer* 16(2):15-24 (1983).

[2] Ammann, P. and Sandhu, R.S. "Extending the Creation Operation in the Schematic Protection Model." *Proc. Sixth Annual Computer Security Applications Conference*, 340-348 (1990).

[3] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Unified Exposition and Multics Interpretation." MTR-2997, Mitre, Bedford, Massachusetts (1975).

[4] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy* 184-194 (1987).

[5] Cohen, E. and Jefferson, D. "Protection in the Hydra Operating System." *5th ACM Symposium on Operating Systems Principles*, 141-160 (1975).

[6] Davies, D.W. "Protection." In Lampson, B.W., Paul, M. and Siegert, H.J. (Editors). *Distributed Systems: An Advanced Course.* Springer-Verlag, 211-245 (1981).

[7] Dennis, J.B. and Van Horn, F.C. "Programming Semantics for Multiprogrammed Computations." *Communications of ACM* 9(3):143-155 (1966).

[8] *Department of Defense Trusted Computer Systems Evaluation Criteria.* DoD 5200.28-STD, Department of Defense National Computer Security Center (1985).

[9] Gong, L. "A Secure Identity-Based Capability System." *IEEE Symposium on Security and Privacy*, 56-63 (1989).

[10] Harrison, M.H., Russo, W.L. and Ullman, J.D. "Protection in Operating Systems." *Communications of ACM* 19(8):461-471 (1976).

[11] Harrison, M.H. and Russo, W.L. "Monotonic Protection Systems." In DeMillo, R.A., Dobkin, D.P., Jones, A.K. and Lipton, R.J. (Editors). *Foundations of Secure Computations.* Academic Press, 337-365 (1978).

[12] Lampson, B.W. "Protection." *5th Princeton Symposium on Information Science and Systems*, 437-443 (1971). Reprinted in *ACM Operating Systems Review* 8(1):18-24 (1974).

[13] Levy, H.M. *Capability-Based Computer Systems.* Digital Press (1984).

[14] Minsky, N. "Synergistic Authorization in Database Systems." *7th International Conference on Very Large Data Bases*, 543-552 (1981).

[15] Mullender, S.J., van Rossum, G., Tanenbaum, A.S., van Renesse, R. and van Staveren, H. "Amoeba: A Distributed Operating System for the 1990s." *IEEE Computer*, 23(5):44-53 (1990).

[16] Sandhu, R.S. "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes." *Journal of ACM* 35(2):404-432 (1988).

[17] Sandhu, R.S "Transformation of Access Rights" *IEEE Symposium on Security and Privacy*, 259-268 (1989).

[18] Sandhu, R.S. "Undecidability of Safety for the Schematic Protection Model with Cyclic Creates." *Journal of Computer and System Sciences*, to appear.

[19] Siberschatz, A., Peterson, J., and Galvin, P. *Operating System Concepts.* Addison Wesley(1991).

EMPLOYEE PRIVACY AND INTRUSION DETECTION SYSTEMS: MONITORING ON THE JOB

Lorrayne J. Schaefer[1]
The MITRE Corporation
7525 Colshire Drive M/S Z 268
McLean, VA 22102

Abstract

The area of intrusion detection systems and privacy has always had a conflict of interest. Intrusion detection systems are designed to help the System Security Officer detect malicious or unauthorized use of a computer system by both unauthorized and authorized users. These systems protect our computer systems from abuse, yet in doing so, it violates our privacy. This paper discusses the legal and ethical issues involved in using an intrusion detection system to monitor the computer system.

Introduction

> Down in the street little eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining and the sky a harsh blue, there seemed to be no colour in anything except the posters that were plastered everywhere. The black-mustacio'd face gazed down from every commanding corner. There was one on the house front immediately opposite. BIG BROTHER IS WATCHING YOU, the caption said, while the dark eyes looked deep into Winston's own...
>
> Behind Winston's back the voice from the telescreen was still babbling away...The telescreen received and transmitted simultaneously. Any sound that Winston made, above the level of a very low whisper, would be picked up by it; moreover, so long as he remained within the field of vision which the metal plaque commanded, he could be seen as well as heard. There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system, the Thought Police plugged in on any individual wire was guesswork. It was even conceivable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to. You had to live - did live, from habit that became instinct - in the assumption that every sound you made was overheard, and, except in darkness, every movement scrutinized [1].

George Orwell's *1984* [1] presents a shocking view of a future where everyone's behavior is carefully scrutinized. The feeling that "Big Brother is watching you" is clearly as unsettling now as it was in 1949, and yet intrusion detection technology now allows computer systems to be monitored by electronic "Big Brothers." This raises many legal

[1] This paper reflects work performed while Ms. Schaefer was an employee of Trusted Information Systems, Inc.

and ethical questions as to exactly what privacy rights employees have, and what lengths companies can go to ensure the security of their computer systems.

Computer security is required for enforcing privacy laws. "At the same time, the process of detecting threats, vulnerabilities and abuses may result in violations of privacy and other human rights, leading to a conflict between the use of computer security to guarantee privacy and its use to invade privacy." [2] One area where this conflict is obvious is in the use of intrusion detection technology. This paper will discuss the legal and ethical issues associated with the use of intrusion detection technology in the work place.

Definitions

Intrusion detection systems (IDS) are System Security Officer (SSO) tools, which aid in the identification of malicious or unauthorized use of a computer system by normal system users (insiders) and unauthorized users (outsiders). In other words, the IDS is used to monitor the computer system.

Intrusion detection systems usually get information from raw audit data retrieved from the observed operating system. Typically, the audit data is then reduced for ease of use. This reduction may involve searching for audit records corresponding to specific events that have been previously deemed important, or simply reorganizing all of the audit records into a more generalized format and disposing of fields that are not needed for further analysis.

The raw content of the audit trail may be system accounting information as well as security relevant events. Generally audit records contain such information as subject (e.g., terminal user, process running on behalf of user), object (e.g., file, device), action performed, time stamp, resource measures, indication of any uses of privilege, and an error code. Most intrusion detection systems are designed to observe abnormal patterns of system use such as failed login, unusual user performance (perhaps an unauthorized user masquerading as a legitimate user), Trojan horses, viruses, or an insider attempting to access unauthorized files [3].

Privacy is extremely important to people, yet its meaning, especially for policy purposes, is often unclear. Privacy represents concerns about autonomy, individuality, personal space, solitude, anonymity, and a host of other related concerns [4]. There have been many attempts to define a "right to privacy." Warren and Brandeis defined it as "the right to be let alone."[5] Webster's dictionary defines it as "one's right to freedom from unauthorized intrusion." Dean Prosser wrote that privacy is "in one form or another...declared to exist by the overwhelming majority of the American courts." [6] Prosser identified four types of privacy invasions: intrusion, disclosure, false light, and appropriation. Each of these types depends on physical invasion or requires publicity, and thus offers minimal protection for privacy of personal information.

The Privacy Act of 1974 protects personal data collected by the government. Any individual can request what data has been collected on him/her, for what purpose, and to whom such information has been disseminated. An additional use of the law is to prevent one government agency from accessing data collected by another agency for another purpose. The Privacy Act requires diligent efforts to preserve the secrecy of private data collected [7].

Webster's Dictionary defines *ethics* as "the discipline dealing with what is good and bad and with moral duty and obligation; the principles of conduct governing an individual or a group."

Of course, "good" and "bad" cannot be precisely defined, since they are relative terms that refer in many cases to personal opinion. Consider two co-workers Jim and Mike. Jim does not think it is wrong to take office supplies:

"I am just taking some pens and floppy disks. It's not going to break them."

"It probably won't," Mike replied, "but it's still wrong. It's company property."

Jim did not think this was wrong, but many others feel it is. We are taught in school, by our parents and by our peers that it is morally wrong to take things that do not belong to us; yet many of us still take "a few pens and pencils."

This is also true with monitoring people on the job. Some think it is acceptable to monitor others because it informs individuals as to who is doing their job properly. Others feel it is only acceptable if there is suspicion that a job is not being properly done. Still others feel that any surveillance at all is ethically wrong. The ethics of what should or should not be monitored is discussed later.

The Use of Intrusion Detection Systems and Privacy Rights

The Privacy Act of 1974 made the individual's right to privacy both a legal and ethical issue. There is an ongoing debate now over where an individual's right to privacy ends and a company's right to protect itself begins.

The use of IDS in the workplace has both advantages and disadvantages. A significant advantage is that it can help detect outsiders breaking into the computer system. It can also help detect insiders abusing company resources (e.g., using company time to develop software for personal profit or committing insider fraud or abuse). Monitoring can be quite useful in environments that have little or no protection of sensitive information, in that an intrusion detection system can help detect unauthorized access to the sensitive information. Some disadvantages employee monitoring can create are low employee morale, reduced productivity, destructive countermeasures, and resentment [8], [9]. While security officials or management may believe monitoring the system protects both individual data and company resources, (i.e., it is not meant to watch over the "good guys" but rather to keep the "bad guys" out) programmers, system developers, and other users of the system may feel that they are automatically an "under suspicion" employee. A middle-of-the-road approach states that if IDS operators were carefully restricted and administered, "monitoring of computer activity could be viewed as a benefit by the user community in the same way as security monitoring of luggage at airports is viewed as a benefit by air travellers" [2].

Monitoring on the Job

An example that makes the dilemma between individual and company's rights painfully clear was published in *Information Week* [10] and the *Washington Post* [11]. Alana Shoars was fired from Epson America, Inc. when she questioned management about its monitoring and reading of electronic messages between employees[2]. There is a question of whether this is a violation of the employees' right to privacy. In 1986, the Electronic Communications Privacy Act (18 U.S. Code 2511) was passed to protect users of telephones and other communications equipment from wiretapping and similar invasions of privacy. The Act also included electronic mail (E-mail), cellular phone service, and other new forms of electronic communication. The Act also extended to communications other than those carried over public networks. It is not clear, however, what rights companies have to monitor the traffic on internal E-mail networks.

There is little question that, at least in the United States, monitoring people without good reason is regarded as socially and ethically unacceptable. Nonetheless, many users of computer systems regard their use of the computer as a personal matter, and a system that watches over their activity could be seen as a violation of privacy. Ironically, people do accept video cameras in banks, airports, and hallways at the workplace. Also, in a shared computing environment, all but novices know that "private" files are not truly private; unscrupulous system administrators and users can examine any cleartext file, and in some cases may be able to read encrypted files. Thus, users generally do not maintain sensitive Privacy Act information on shared systems that lack adequate protection measures. Perhaps the main reason intrusion detection systems appear threatening is that they are designed to judge user activity, specifically to determine whether or not a user is behaving normally or violating some security policy [13].

What to Monitor?

An audit or intrusion detection tool is designed to detect anomalous behavior. Generally, it is intended to aid the SSO in locating the "bad guys" who are circumventing the system. But what about the "good guys"? Exactly how much system activity should an intrusion detection system monitor? In other words, when does this start going beyond a tool and begin invading someone's right to privacy? The IDS will not invade a person's privacy rights if it is monitoring at the node level (login failures). If the IDS is monitoring every keystroke of an individual, this would be an example of invading a person's privacy. On the other hand, the tool could point out that an individual has been poking around files to which she has no access rights. The SSO can then take preventive or preemptive action. There certainly are enough cases of employee fraud where extensive auditing would be deemed not only appropriate but imperative by management. Financial institutions could hardly expect to be insured if a strong audit program were not in place.

As mentioned earlier, people are monitored all the time -- in airports, banks, supermarkets, and department stores, to name a few common places. This usually does not upset people. It is expected that a camera will monitor activity in these areas to help protect both the public and the company assets, as well as to offer a warning to potential trouble-makers.

But what about being monitored on the job? All forms of surveillance and supervision are accepted in factories. Factory workers owe 100% work time when they are on the job in the factory. Whatever workers build in the factory is the factory's property.

---

2 This case was dismissed January 1991 by a superior court judge who said the California wiretapping statute does not apply to E-mail. That suit, however, is pending appeal [12].

This should also be true with white-collar jobs. All scientific discoveries made at work are the company's property. Employees should not spend company resources making several personal calls or revising résumés [14].

There is, however, an unspoken ethic that it is morally wrong to rifle through fellow employee's drawers or files, or eavesdrop on phone conversations.

An employee can consciously protect her files from being monitored. She can do this by either calling an important document something meaningless or by putting file protections on the document to prevent wandering eyes from seeing it. Even so, it is well known that these hurdles can be brought down with little or no effort. The difference between these examples and the knowledge that your system is being monitored by an IDS is that in the former scenario the employee is still comforted with the unknown -- she really is not sure that she is indeed being monitored. The latter case can change employee behavior with the knowledge of being monitored.

As described earlier, employee monitoring may result in low employee morale, reduced productivity, destructive countermeasures, fear, and resentment. As a real-life example, take the boss who automatically has a file sent to him each time someone first accesses their electronic mail. The boss uses the time stamp to determine when that employee has arrived for work; that is, if the employee reads his mail as soon as he arrives. This is a classic example of organizations analyzing patterns of E-mail. Employees can, of course, purposely not read their mail until the afternoon.

As another example, suppose your supervisor, John, approaches you and asks why you can't do an additional task to those currently assigned. You tell John that you don't have enough time. Without your knowledge, John starts monitoring your daily activities using an IDS. John notices that you spend more time reading personal mail than you should. John approaches you later and accuses you of spending an average of two hours reading mail per day and that if you spent less time reading personal mail, you would have plenty of time to do the additional task. How would you feel in this situation? Most people would probably be outraged, resenting the fact that John monitored them without prior permission.

Even if employees know that extensive intrusion detection systems are used, the two examples above illustrate the use of monitoring tools being abused by the unethical. The examples illustrate extreme uses of intrusion detection systems in a "Big Brother is watching" fashion.

Conclusions

Yet examples such as these happen often in the workplace. Using an IDS to monitor the system is an excellent tool to aid the SSO in detecting attempted system breakins or employee abuse of company resources. But this tool also makes it possible to abuse moral issues such as spying on individuals or using the IDS to calculate employee performance. A company using intrusion detection systems must face many legal and ethical questions that to date have not been completely answered. Thus, each organization planning to use such a system should consider these issues.

There are at least two major legal issues that need to be identified. First, whether or not companies have the legal right to monitor computer use, and second, at what level could such monitoring occur. Companies demand the right to monitor computer use to protect proprietary information and to prevent abuse of computer resources. Companies should have a written policy that describes the extent to monitoring the system.

Even though the legal issues are not well-defined today, these issues should be better understood in the near future. With the *Shoars v. Epson* E-mail case, many companies are becoming more aware of the legal and ethical issues. This case has prompted many organizations to review their policies on system security and employee monitoring, and some companies that previously had no policy on system monitoring have created one.

Companies who do not have a policy could have problems if they have to go to court to defend themselves concerning the monitoring of employees. It is clearly wise for companies to develop a policy regarding the use of IDS. The policy should cover issues such as limits of IDS use, use of the results obtained from monitoring, obtaining informed consent of users, and providing due notice of intent to monitor. The development of this policy should not be limited to security experts, but should involve system users, as well as psychologists, sociologists, constitutional lawyers, and human rights groups [2]. This security policy should be openly available to employees. Each employee should read and sign the policy indicating that they understand and will abide by the rules within. Employees should be advised that they are being monitored when they are using company computing resources. It should be very clear as to what exactly is being monitored and how that information will be used.

Bad policy can certainly become a reality within a company's use on intrusion detection systems. There is also a possibility that the IDS operator can abuse the tool to monitor anything and everything employees do, thereby becoming a kind of Big Brother. Who should or shall oversee that companies do not, in fact, abuse this technology, which is otherwise a great benefit for information security, should be explored to prevent the workplace from being under the constant surveillance of Big Brother and the Thought Police.

References

[1] Orwell, George, *1984*, Harcourt, Brace, and Company, Inc., NY, 1949, pp 6-7.

[2] Denning, Dorothy E., Peter G. Neumann, and Donn B. Parker, "Social Aspects of Computer Security," *Proceedings of the 10th National Computer Security Conference*, September 1987.

[3] Schaefer, Lorrayne J. and J. Noelle McAuliffe, "Intrusion Detection Technologies," Trusted Information Systems Technical Report #334, February 1990.

[4] Federal Government Information Technology, *Electronic Record Systems and Individual Privacy*, OTA-CIT-296, U.S. Office of Technology Assessment, Washington, D.C., June 1986.

[5] Warren, S.D. and L.D. Brandeis, "The Right to Privacy," *Harvard Law Review*, December 1890, pp 193-220.

[6] Prosser, Dean, "Privacy," *California Law Review*, vol. 48, 1980, pp. 383, 386.

[7] *Privacy Act of 1974*, Public Law 93-579, U.S. Code 552(a), December, 1974.

[8] Irving, R.H., C.A. Higgins, and F.R. Safayeni, "Computerized Performance Monitoring Systems: Use and Abuse," *Comm. ACM*, August 1986, pp 794-801.

[9] Marx, Gary T. and Sanford Sherizen, "Monitoring on the Job: How to Protect Privacy as well as Property," *Technology Review*, November/December 1986, pp 63-72.

[10] Caldwell, Bruce, "Big Brother is Watching," *Information Week*, June 18, 1990, pp 34-36.

[11] Richards, Evelyn, "Privacy at the Office: Is There a Right to Snoop?," Business Section, *Washington Post*, September 9, 1990, pp. H6, H8, H9.

[12] Eckerson, Wayne, "E-mail Privacy Issue Gains Momentum With Second Suit," *Network World*, February 11, 1991, p. 33.

[13] Bauer, David S. and Dorothy E. Denning, "Social and Privacy Issues of Intrusion Detection," *Proceedings of the 1st SRI Intrusion Detection Workshop*, March 1988.

[14] Garson, Barbara, *The Electronic Sweatshop: How Computers are Transforming the Office of the Future into the Factory of the Past*, Simon and Schuster, New York, NY, 1988, pp 205-224.

EXPERIENCE OF COMMERCIAL SECURITY EVALUATION

Peter Fagan & Julian Straw
Secure Information Systems Limited, Sentinel House, Harvest Crescent, Ancells Park, Fleet, Hampshire, GU13 8UZ, England

ABSTRACT

Considerable experience has been gained in Government funded or controlled facilities in the United States, in the UK and elsewhere in the evaluation of systems and products. This paper discusses experiences gained from the operation and management of a UK Commercial Licensed Evaluation Facility (CLEF), and highlights the issues involved in the marketing of certification to security product vendors.

INTRODUCTION

Two licensed commercial evaluation facilities have been operating in the UK since June 1989. The contracts to operate the CLEFs were granted to two parent companies, Logica Space and Defence Limited and Secure Information Systems Limited (SISL), as the result of a competitive tender process.

In contrast to the existing UK Government funded evaluation facilities, the CLEFs operate on a commercial basis, seeking evaluation work from product suppliers and project sponsors. The CLEFs have now been in operation for two years and have provided unique experience in the area of commercially funded formal evaluations. This paper outlines the procedure for conducting such evaluations, and discusses the issues raised under headings of licensing, staffing, management and marketing. The views expressed are those of the SISL CLEF only.

CONDUCT OF EVALUATIONS

Evaluations are carried out in the SISL CLEF against the evaluation criteria developed by the UK Communications Electronic Security Group (CESG) [1] and also the harmonised European IT Security Evaluation Criteria (ITSEC) [2]. The UK criteria define levels of assurance, describing how confidence is obtained in the design, implementation and operation of a product or system, but without applying constraints to the functionality of any item submitted for evaluation. Because of this the criteria can be applied to systems and products with limited and specific features but which meet high assurance requirements. The ITSEC are compatible with the approach in [1], while at the same time being designed to provide a link to evaluations of products using functionality as defined in the DOD Trusted Computer System Evaluation Criteria (TCSEC) [3]. The ITSEC are the result of an initiative by the UK, France, Germany and the Netherlands, and are based on existing European criteria.

Because functionality and assurance are split, a target evaluation level (e.g. ITSEC E3) is insufficient in itself to define the duration and extent of an evaluation. The information required to control a UK evaluation is provided in two documents: the evaluation baseline and the evaluation work programme.

The baseline document defines the scope of the evaluation work. It states the target assurance level but also states the extent of the functionality of the item under evaluation. For ITSEC evaluations this statement will either refer to or contain a Security Target, which in the case of a system will describe the security policy for the system. In the case of a product the Security Target will contain a set of claims, describing the security features provided by the product, and a rationale which enables a prospective purchaser to assess whether the product will meet his requirements.

The work programme lists all the work packages making up the evaluation, and states the amount of effort assigned to each.

The baseline and work programme are issued in parallel and both are approved by the certification body before the commencement of an evaluation. This ensures that necessary and sufficient work is planned to allow the product to be evaluated, given the functionality claimed for the product, and that the quality of any evaluation remains unaffected by the competitive situation.

Evaluation then proceeds according to the requirements stated in the ITSEC or the UK criteria, guided by an evaluation manual issued by the certification body, which provides a rationale for standard work packages and describes contents and layout for the mandated reports.

This paper discusses the commercial nature of the relationships between the parties involved in such an evaluation and the issues raised in operating such a facility on a commercial basis. Figure 1 shows the major parties involved in a UK commercial product evaluation and illustrates the flows between them. The roles of these parties are described fully in the UK IT Security Evaluation and Certification Scheme Publication No 1 [5].

The relationship between the facility and the certification body requires commercially sensitive information to be passed from the facility. This is to enable the certification body to monitor progress and assist with the resolution of any problems which arise. The certification body also refers government projects with a requirement for evaluation to the evaluation facilities, in an equitable manner. In the UK both the commercially operated facilities and the certification body are committed to the success of the CLEF scheme and work in concert, given the constraints of their different roles.

LICENSING

On 1st May 1991, the single UK IT Security Evaluation and Certification Scheme was launched in the UK, replacing the scheme under which the CLEFs had been initially established. The new scheme is managed jointly by CESG and the UK Department of Trade and Industry (DTI), under the direction of a management board made up of representatives of a number of UK Government departments. The scheme provides for a single UK Certification Body, reporting to the board.

The SISL CLEF is licensed under the scheme, to carry out evaluations using the methodology common to all UK evaluation facilities. The licence is granted by the certification body under the terms described in UK IT Security Evaluation and Certification Scheme Publication No 2 [6].

The terms of the licence place constraints on the operation of the facility in the areas of security procedures and management. In particular it is a requirement that a quality system which has been accredited by the UK National Measurement Accreditation Service (NAMAS) be in place at the facility. This fact is appreciated by CLEF clients since it increases their confidence in the quality of the work performed. NAMAS is a service operated by the UK National Physical Laboratory (NPL). The criteria used by NAMAS assessors are primarily reliability, quality and traceability of results. The certificates awarded by NAMAS are recognised widely within the UK, and mutual recognition agreements are in place with a number of European countries.

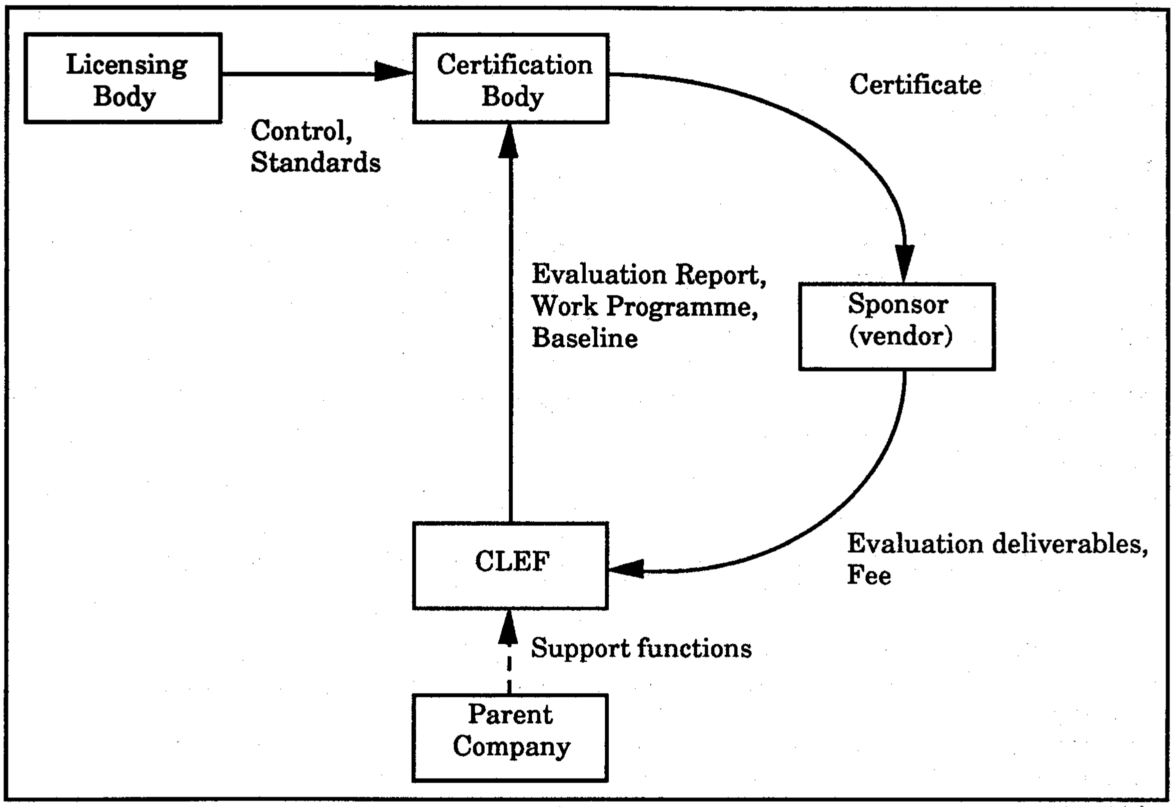


Figure 1 : Commercial Relationships

The licensing terms also require an appropriate management structure to be in place. In general terms this comprises a facility controller, responsible for the overall operation, a business manager (reporting to the facility controller and therefore keeping control on commercially sensitive information), a technical manager (responsible for day to day operation), and an administration manager (whose responsibilities lie mainly in the area of day to day security). In addition there are potentially a number of specialist roles such as methods advisor (responsible for advising on the use of formal methods and associated static and dynamic analysis tools). While it is possible for one individual to hold more than one of these posts, two other posts exist which act as an internal check on the operation of the facility, and which therefore cannot be combined with any of the other roles. These are the posts of quality manager and security manager. It is an important aspect of these two positions that they are independent of the facility controller, in order to assure their impartiality.

The licensing terms for the UK facilities ensure very high standards of work, and the standards are coupled with an official endorsement of the work carried out. Unlicensed companies offering similar services may in some cases be able to undercut the facilities in terms of price; however their work will not carry the authority required for a vendor to achieve the desired marketing benefits provided by a government certificate awarded after evaluation in an approved facility.

STAFFING

In addition to licensing the facilities, CESG operate a separate scheme through which individual evaluators are licensed on the basis of their training and experience. Training courses can be run by a facility subject to approval from the certification body, which reviews the content and quality of the course. This again increases customer confidence and allows the licensing body to ensure that suitably qualified staff are used on evaluations.

The requirement for licensed staff creates a problem for a commercially-operated facility, where the flow of work may be irregular. Training and licensing of individuals constitutes an investment which must be used in the facility if it is to bring benefits. Therefore at the end of an evaluation, when qualified staff potentially become unassigned, there is a need to retain them in the CLEF and not to return them to the parent company, where they may be assigned to long term projects and thus become unavailable to the facility. A stable and self-contained community of evaluators is to be desired. However, only limited overheads in terms of low staff utilisation levels can be tolerated in a commercial environment. These two conflicting requirements can only be reconciled if there is a stable and reasonable flow of evaluations into the facility, which in turn requires a commercially-oriented approach to operation and marketing.

In order to minimise costs and risks to the CLEF, personnel are assigned for the duration of an evaluation, and only very exceptionally are they removed for other work within the facility.

The possible consequences of breaches of commercial confidentiality affect staffing. While document security, procedural security and physical security can be adequately addressed by the means usually adopted by defence contractors, personnel security is an area requiring increased attention. Non-disclosure agreements are made on an individual basis, so as to confine the spread of information to those with a need to know. This agreement continues beyond the lifetime of the evaluation. In addition, constraints are placed on the management of facility staff, so that their deployment outside the facility will not place them in positions where they could use information gained during the evaluation to the commercial disadvantage of the vendor. Monitoring of staff who leave the facility and the parent company remains a problem.

Staff motivation is an issue within CLEFs. Since one aim of evaluation and licensing is to achieve standardisation, a danger exists that staff can be left with a feeling of insufficient autonomy. This issue is considered to be an important one within the SISL facility since staff motivation is a prerequisite for high quality work.

Autonomy and feedback on performance are two major factors affecting motivation, secondary issues being task significance and task integrity. Autonomy needs to be a feature of a facility, with early responsibility and customer contact. While commercially desirable technical skills are gained during the evaluation of a product, these tend to be knowledge of the construction of the product rather than knowledge of its use. Also there are constraints on the use of the knowledge gained during the execution of an evaluation. Autonomy can provide experience which compensates for this. Similarly, early and frequent feedback to staff on performance and problems must be a feature of a primarily participatory management style. By incorporating the commercial aspects of evaluation (e.g. proposal preparation, presentations), into all positions in the facility, task significance and task integrity can be achieved. In the experience of the authors this is best carried out by sharing the marketing work and administration tasks.

MANAGEMENT

Commercial confidentiality is a major issue in the management of a facility as well as in the licensing.

The key benefit which vendors see in obtaining certification for a product is that of obtaining a marketing asset. This is particularly true in the case of certification to the ITSEC. In contrast to the TCSEC, increased emphasis is placed on the development environment for a product, successful evaluation reflecting upon the company and its development process just as much as on the product. Therefore the timing of the announcement of certification, and the confidentiality of the results of evaluation are important factors in UK evaluations.

This requirement for commercial confidentiality arises in part from the commercial nature of CLEF work. Knowledge of any corporate action expected shortly to provide an improved market share might be considered by many vendors to be sensitive information. Where longer timescales are expected, as is the case in the US, the early announcement of formal evaluation may be beneficial. Since UK commercial evaluations are conducted with the minimum evaluation effort commensurate with the maintenance of the enforced standards, there are in contrast, potential benefits for a vendor choosing confidentiality.

This means that great care must be taken with the handling of customer identities within the facility, and also within the parent company where facilities such as accounts, sales and marketing are used. Identities of prospective and actual customers are disguised by an internal numbering scheme, with only the minimum number of staff knowing for whom the work is being carried out.

Strict measures are put in place within the facility to provide commercial confidentiality. Primarily this is a matter of physically separating teams working on separate evaluations, and providing secure storage facilities for each. Preferably teams should use dedicated computer equipment which is flushed between evaluations, since working on two evaluations simultaneously on the same computer places an increased dependence on logical separation of user groups.

The SISL facility is physically separated from its parent company, with a separate entrance. This separation reduces the risk of accidental disclosure via documents or conversations. In addition, prospective customers can be seen without the knowledge of personnel in the parent company. A log of visitors to the facility is kept, and managed in a way which prevents one prospective customer from seeing that another has visited. The same is true of any document recording the identity of more than one customer (e.g. facsimile log, business reports). Visitor passes are issued by the facility and not by the parent company so that no record is kept of the visit by the parent company. Separate telephone and facsimile lines are provided so that customers and prospective customers can be assured of the confidentiality of their project.

For all tasks, individual registers are kept of all deliverables supplied to the facility whether or not there is a requirement to handle classified information. At the end of the evaluation the deliverables provided to the CLEF are either returned or destroyed, as agreed between the facility and the client. Appropriate destruction procedures are among those defined by the Security Operating Procedures (SOPs) under which the facility operates. The SOPs, approved by the certification body, define the procedures for day to day management of the facility and for the handling of the client evaluation deliverables. The existence of documented procedures is essential for consistent application of confidentiality and quality requirements.

From the point of view of confidentiality the facility can be seen to comprise a number of operating groups: those with knowledge of prospective customers (the facility controller and the business manager, together with any staff involved with sales support); those with knowledge of a particular evaluation; and those with overall knowledge of the CLEF operations. In fact this last group consists of a single individual, the facility controller.

While it can be seen that there are management problems in operating a CLEF as an arm of a parent company, there are advantages also. It is unlikely, for example, that any one evaluation will be a significant fraction of parent company turnover; therefore cash flow on one task is unlikely to be a significant factor for the overall health of the parent.

The primary problem in the management of CLEF evaluations is one of maintaining control on costs. During the evaluation this is exemplified by the problem of evaluating a product in which minor faults may be found which must be corrected before certification. Clearly this will require some re-evaluation, and a suitable strategy must be chosen during contract negotiation which will allow this to take place within the constraints of what is usually a fixed price contract.

In entering a contract with a vendor the facility is in the unusual position of performing a service without guaranteeing a result, since it does not itself award the certificate. The evaluation results cannot currently be provided directly to the vendor, and the same is true of information concerning faults which may be found in the system or product. These are sent instead to the certification body who can release them (or not) to the vendor. This can lead to situations where vendors may attempt to impose unacceptable constraints on the evaluators, such as penalty clauses in the event of the certification body not responding within defined timescales.

Commercial risks to the CLEF in entering into a fixed price contract are naturally a management issue. The availability of deliverable items such as design documents and source code has an impact on timescales and costs. In order for the evaluation to proceed the deliverables must be provided at an early stage, and it is usual for a contractual clause to exist which will protect the facility in the case of these items being delayed or being unavailable. To guard against the effects of this situation, clear lists of required deliverables are provided to vendors at the time of submission of a proposal by the CLEF.

The commercial liability which a CLEF is prepared to accept is defined by the terms of its insurance cover and by the status of the reports which it produces. The SISL CLEF is covered for example for security evaluations, but not for safety critical uses. The legal status of an evaluation report is that of a statement that the product has been compared against a certain standard; not that the CLEF is guaranteeing the product to be secure. This is obviously essential to protect against third party claims for consequential loss.

A final management issue concerns the use of tools in the CLEF. For a commercially operated facility the use of tools in areas such as source code analysis is justified where a saving will be made or where the quality of the evaluation will be improved. For example at higher levels of assurance it may be necessary to use a tool to achieve the required confidence level. Previously this area has to some extent been one of academic research, and the tools which will be necessary to derive commercial benefit for a CLEF may be different to those which are currently being developed.

PRE-EVALUATION CONSULTANCY

Vendors consider the price of an evaluation as speculative investment, and an internal marketing case may have to be provided before senior management will allocate a budget for an evaluation. Evidence may have to be provided by the technical department that it is confident of a successful result. To meet this need the facility offers pre-evaluation consultancy. The aim of this activity is to highlight areas which should be addressed before committing to an evaluation proper. A review period is sometimes beneficial, so that any corrective action can be verified.

The aims of pre-evaluation consultancy depend on the requirements of the vendor. Frequently the vendor will wish to understand more fully the evaluation process and the risks which are being accepted. To meet this requirement the facility produces a vendor report which describes the deliverables required for a target level, and assesses the available deliverable items against those requirements. It may be that as part of the consultancy, a vendor will authorise an evaluation at a level below the desired level, as a cost-effective check on the evaluatability of the product, or just to confirm the target level. Vendors may also wish to compare the requirements of an NCSC evaluation against those of an ITSEC evaluation, to determine the effectiveness of an ITSEC evaluation in terms of addressing the European market. An important form of pre-evaluation consultancy is in the preparation of a baseline and work programme. The agreement of the certification body is required before an evaluation can commence, and the controlling documents are required before such agreement can be given. Therefore where a CLEF enters into an evaluation contract without having agreed the baseline and work programme, it does so at its own commercial risk. A clear contractual and licensing distinction is drawn between evaluation and this form of consultancy.

Therefore the common aim of all pre-evaluation consultancy can be seen to be to reduce risk in the evaluation phase, both to the facility and to the vendor.

During the management of any form of pre-evaluation work it is important for the CLEF to maintain impartiality. It has been suggested that CLEFs should be debarred from performing evaluations where they have provided pre-evaluation consultancy. The basis of this argument is that a conflict of interest can arise if a facility first determines the suitability of a product for an evaluation which it then subsequently carries out. There are dangers in this view for all parties. It is unlikely that other organisations offering such services will be licensed or policed in the same way as CLEFs, and undoubtedly to protect their commercial interests there will be disclaimers attached to the results. Such consultancy will be provided in the absence of experience of evaluation itself and in the absence of up to date knowledge of the remit of the approved facilities. Most importantly the consultancies will be taking on the role of the CLEF during the period in which a CLEF would be gaining experience in the product and building a relationship with the vendor. If a CLEF were to come in at the evaluation stage without having reduced their own risks beforehand, the net effect would be higher prices for evaluation, reflecting a higher contingency, in the light of possible contractual and quality problems arising from the previous stage.

MARKETING

Evaluation is currently considered to be primarily a vertical market, in that the skills sold by the evaluation facilities are narrow in range and are applied in a similar way to varying sizes of project. However the SISL CLEF has found that skills are gained during evaluation which should allow a broader base to be established, and which would enable evaluation expertise to be applied across a wider range, possibly by applying subsets of the skills (e.g. source code analysis services, secure product design reviews).

There are some unique issues to be addressed in marketing these, and other, CLEF services.

In order to understand the marketing issues in any industry it is useful to split the market into appropriate segments (e.g. by customer type, contract size or geographical area). Segmentation of the evaluation market, and the parameters which could be used, are issues yet to be addressed in detail by the UK evaluation community. The primary reason being that the low level of activity means that there is a limited amount of information to gather and thus analyse. However it will soon be necessary for answers to be supplied to questions such as 'how is the market split?' and 'how does our CLEF expertise map onto that split - what market share will that give us?'. Initially however it can be assumed that at higher levels the evaluation market is predominantly for certification of products for use in Government systems. The price of evaluation for these products may be considered by vendors to be the price of admission to the market.

Publicity following certification is another marketing issue for vendors. Press releases are effective to a degree in alerting the public to a product undergoing evaluation, and in the UK this has been employed at the lower end of the market. This is useful for the facilities since it also alerts other vendors to the expected benefits. However, the facilities are still bound by their agreements on confidentiality and this is a constraint on their marketing operations.

Currently the UK market for evaluation is a latent one (the market is considered to have potential but currently it is not running at a very high level of activity). In these circumstances a pro-active approach is required to identify and stimulate market areas. This is made more important to the CLEFs by the vendor requirement to reduce through-life costs. In short, when a vendor has undertaken an evaluation with a facility, and the quality of the original work has met expectations, the staff of that facility will have been trained in the design of the product. It will make commercial sense for the vendor to return to the same facility for subsequent re-evaluations. Thus it is important from the point of view of the facilities to be pro-active, since repeat business is generally considered to be cheaper to obtain than new business, and since there are a finite number of product vendors.

In a competitive environment a CLEF must decide on a marketing stance. Although performing the original work has been stated to be a factor in winning repeat business, it can be expected also that the so-called 'marketing approach' will be the optimum stance in the long term. In this the CLEF seeks to understand specific vendor requirements and to match the work to the requirements, providing customer satisfaction as an internal goal. In a field in which it is clearly possible to provide a service on a basis of 'take it or leave it', the marketing approach can be expected to provide distinction to a CLEF.

Publication of the ITSEC has undoubtedly raised awareness of the process of certification. The harmonised European criteria are increasingly relevant to the marketing activities of European subsidiaries of US companies. When the planned framework for the ITSEC has been put in place, the validity of evaluations will be accepted throughout a number of countries, with the evaluation scheme recognising fully the TCSEC functionality which many vendors will have incorporated. The use of the ITSEC will provide a number of benefits for commercial evaluation facilities, primarily removal of the requirement for published interpretations of the criteria in particular circumstances or for particular applications. The UK facilities have not for example been delayed in database evaluations by the absence until recently of an accepted version of the Trusted Database Interpretation [4]. However, in comparison to the US market the UK market is small, and therefore the flexibility of criteria such as the ITSEC has not been exploited on a scale necessary to achieve significant marketing benefits as yet for the facilities.

The issue of cost-benefit analysis as performed by a vendor must be considered. It has already been stated that in simple terms a vendor will see evaluation costs as a speculative investment which can be expected to reap rewards in terms of increased sales.

This is nowhere more true than in situations where a vendor is aiming for a low level of assurance in a simple product. A statement of assurance gained by an approved facility in the correctness of a product, coupled with a statement from the vendor of his security claims, provides a marketing asset sufficient to distinguish a product from its competitors. The total cost of evaluation for a Personal Computer (PC) security add-on at the lowest assurance level might be in the region of $7,000-$12,000. The elapsed time would be a matter of a very few weeks. Therefore even distributors (rather than manufacturers) can and do consider this as a feasible investment. For a comprehensive PC security product, moving up to a higher level of assurance could cost $40,000-$60,000 and would run for perhaps 12-15 weeks. This is a different sector of the market, and different reasons for acquiring certification apply.

MAINTENANCE OF CERTIFICATION

Aside from the initial costs, through-life costs are an issue for vendors, and are therefore a marketing issue for CLEFs. For product vendors at any level there is an overriding commercial benefit in being able to control the costs of certificate maintenance. It may for example prove more cost-effective for the vendor to decouple a certificate maintenance programme from the usual product release cycle. The ITSEC take into account the quality of the development environment, and a vendor may be content to run through two or three bug fixes or releases before going for recertification, relying on customer confidence in the certification of the development environment. Strong configuration control requirements for a product enable the impact of changes on product security to be closely monitored and assessed. A commercially acceptable scheme for maintenance of certification will provide control to the vendor over the timing and size of expenditure.

The separation of assurance from functionality has allowed small companies to gain certification for simple products at low levels of assurance. Maintenance of certification at these levels is generally accepted to be almost a re-evaluation. The costs of maintaining certification as the product evolves will probably be significant in terms of the vendor company turnover. Nonetheless, in the UK scheme the costs and timings for re-evaluation are under the control of the sponsor. There is no requirement for open ended support or liaison with the evaluation facility or certification body, which would be inappropriate for a small organisation.

For larger products or systems, the nature of re-evaluation is different. Work must be done during the evaluation to allow the certification maintenance to proceed, by constructing a database of security relevant areas. After certification, changes are notified to the evaluators by the vendor, and the evaluators assess the impact on certification, providing where necessary, third party evidence that certification remains unaffected. However at this level also the costs and timings of re-evaluation remain under the control of the sponsor.

SUMMARY

The relative advantages and disadvantages of commercial security evaluation are summarised in Figure 2 below.

The market for commercial evaluation in the UK remains in its infancy. The UK CLEF scheme has established a model for its development which has now been tested and refined, and which will be sufficient to meet an expanded market. The ITSEC provide a widely applicable basis upon which to build, and are expected to generate significant interest from international markets in the use of UK facilities.

The establishment of the CLEFs has required considerable effort in the definition of procedures for licensing and certification. It has also called for the resolution of problems concerning commercial confidentiality and staffing. Market development has called for careful examination of the requirements and motives of potential customers, to determine how best their needs may be satisfied.

It is now believed that the UK model for commercial evaluation facilities, with its emphasis on quality management and responsiveness to market needs, can provide the basis for a significant expansion in the supply of evaluated products available internationally.

	Advantages	Disadvantages
Commercial (CLEFs)	reduced timescales no queues reduced client restrictions adaptable to customer needs	cost to vendor not well known in US
Government (NCSC)	free widely known	long timescales not development env only US systems queues

Figure 2 : Comparison Summary

REFERENCED DOCUMENTS

1. CESG Computer Security Memorandum Number 3
   UK Systems Security Confidence Levels, Issue 1.1, February 1989

2. Information Technology Security Evaluation Criteria (ITSEC)
   Harmonised criteria of France, Germany, the Netherlands, the UK
   Version 1, dated 02 May 1990

3. Department of Defense Trusted Computer System Evaluation Criteria,
   DoD 5200.28-STD, December 1985

4. Trusted Database Management System Interpretation of
   Trusted Computer System Evaluation Criteria, NCSC-TG-021, Version 1,
   April 1991

5. UK IT Security Evaluation and Certification Scheme Publication No 1
   Description of the Scheme
   Issue 1.0, dated 1 March 1991

6. UK IT Security Evaluation and Certification Scheme Publication No 2
   The Licensing of Commercial Licensed Evaluation Facilities
   Issue 1.0, dated 1 March 1991

EXPERIENCES IN MULTI-LEVEL SECURITY ON DISTRIBUTED ARCHITECTURES

Karl A. Siil
AT&T Bell Laboratories
1 Whippany Road, Rm 14E-218
Whippany, New Jersey 07981

ABSTRACT

This paper describes the port of a Multi-Level Secure (MLS) operating system to a multi-processor distributed architecture. The implementation involved porting AT&T's B1 Rated System V/MLS to the AT&T 3B4000 super-minicomputer. Although originally a port of the System V/MLS *operating system*, the 3B4000 port provided valuable experience in solving problems associated with MLS networking. This type of experience is required for the creation of future secure system solutions. Because, just as it is unlikely for a modern computer not to be networked, it is equally unlikely for an MLS computer not to have need of MLS networking.

INTRODUCTION

This paper describes the port of AT&T's System V/MLS to the AT&T 3B4000 super-minicomputer. During this port, the 3B4000 distributed architecture's resemblance to a network unexpectedly provided answers to and performance data on MLS networking issues that were not part of the original goals of the porting project.

MLS networking is a relatively new and unexplored territory that is in demand by customers in both the government and commercial realms. Theoretical pursuits, although constructive for determining avenues of endeavor, are limited by the many real-world issues that, as yet, cannot be expressed as equations, proofs, or algorithms. Worked examples of MLS networking are needed to confront and resolve such issues. The System V/MLS 3B4000 port has acted as one such worked example.

The paper starts with brief overviews of System V/MLS, the 3B4000 distributed architecture, and UNIX®† System V on the 3B4000 concentrating on those elements that are relevant to multi-processor and network security. It then goes on to present a set of porting requirements determined before starting the port. The paper then discusses network security issues encountered and tackled during the port. It goes on to discuss the impact of the requirements and network security issues on the System V/MLS kernel modules, commands, and libraries. The paper then shows how achievement of the requirements was verified. Lastly, the results are extended to network security in general showing how the porting experience can be applied to the development of MLS networking products.

SYSTEM V/MLS AND 3B4000 OVERVIEWS

This section presents overviews of System V/MLS and the 3B4000 distributed architecture. This is to provide the reader with the basis for the discussion of what was undertaken in the port and how the results of the port can be applied to MLS networks.

System V/MLS

System V/MLS (SV/MLS) is an NCSC B1 Rated version of the UNIX System V operating system. The first

† UNIX is a Registered Trademark of UNIX System Laboratories.

UNIX system to achieve a B1 Rating, SV/MLS is fully compliant with the System V Interface Definition (SVID) and introduces no more than 4% performance degradation with full auditing enabled. For this paper, the most important components of SV/MLS are label management and audit subsystems.

The SV/MLS Mandatory Access Control (MAC) policy is a modified version of the policy described by Bell and LaPadula.[1] Subjects and objects are labeled via an overloading of the conventional UNIX GID field. Unlike conventional UNIX systems, this field is not a dimensionless number on SV/MLS. Instead, it is an index into a file. Each element in the file is composed of the data necessary to form an SV/MLS *privilege.* A privilege can be thought of as an instance of a conventional UNIX group at a given MAC label. An example of two privileges is shown in (**Figure 1**).

Privilege = Discretionary Group + MAC Label

For example, in the *ls*(1) output:

```
-rw-r-----   1 karl    X[TS]      31056 Jun 11 13:19 Mission_Data
-r--r-----   1 karl    X[S]       58547 Apr 17 11:03 Design_Doc
```

The privileges, `X[TS]` and `X[S]` could be:

Privilege	Group	Label
X[TS]	Project_X	Top Secret
X[S]	Project_X	Secret

Figure 1. System V/MLS Privilege Components

When an access check is performed, privilege information is required by the access control software. SV/MLS uses a cache to hold frequently referenced privileges. The cache is checked before any privilege information is read from the disk file. The file and cache have become known as the *labels file* and *labels cache*, respectively. Technically, both contain more than labels. But, to maintain a smooth flow in the paper, the commonly used terms shall also be used here. Also, the act of getting information from either of the labels cache or labels file has become known as, *getting a label.* That terminology will be used here, as well.

There are three (3) components to the SV/MLS audit subsystem. These are:

- A kernel-resident audit trail data buffer,
- An audit trail daemon process, and,
- A set of audit trail data files.

The SV/MLS audit trail is composed of binary records collected from 25 probe points throughout the kernel, as well as 16 trace devices accessible only by trusted processes (e.g. *login*(1S), *passwd*(1S), etc.) via nodes in /dev. The binary data is sent from the probe points and trace devices to the kernel buffer. The buffer is periodically read by the trusted daemon process which then sends the audit data to the files.[1] When a System Security Officer wishes to review the audit data, he/she passes the binary data through a filter program to convert the data to a human-readable format.

1. The audit trail daemon can write to *any* form of writable media (files, printers, network ports, etc.). Most SV/MLS sites use files, however. So, the remainder of this paper only discusses files as targets for audit data.

AT&T 3B4000 Distributed Architecture

The 3B4000 super-minicomputer is a multi-processor system composed of up to 16 Processing Elements (PE's). The system is composed of a 3B15 Master Processor (MP) with up to 15 Adjunct Processing Elements (APE's) physically connected by a network fabric known as the A-bus.[2] Though the hardware is a star topology, the 3B4000 kernel software supports logical point-to-point and broadcast type messaging between PE's. Each PE runs its own UNIX kernel. The APE's depend on the MP to provide services for many operating system functions (e.g., process creation, clock synchronization) and cannot run autonomously for any great length of time. However, each APE maintains data structures for its local files, inodes, mounted file-systems, devices, etc. These data structures are similar to those of uniprocessor UNIX systems.

When a PE requires a resource/service from another PE (e.g., a remote file access), it issues an A-bus message requesting that service. If the server PE can satisfy the client PE's request immediately, the client waits for the results. If not, the client process requesting the information goes to sleep and gives control of the CPU to the next runnable process. When the information requested from the server arrives, the requesting process is made runnable and, when given the CPU, it retrieves the data and continues its work.

One main goal of the designers of the 3B4000 version of UNIX System V was to maintain application compatibility at the object code level. The internals of many system calls were augmented to handle the need for sending and/or receiving messages over the A-bus. But, most of these changes are invisible to applications programs.

PORTING REQUIREMENTS

Before the SV/MLS 3B4000 port began, several documents were referenced to determine the porting requirements. The three (3) primary sources used were: The Orange Book,[3] the uniprocessor SV/MLS Design Documents,[4] [5] and the 3B4000 UNIX System V Design Documents.[2] Using these sources and a few others, requirements were determined for the SV/MLS 3B4000 port. In the list that follows, the requirement itself is in *italics*. Additional information is provided to explain the requirement and its purpose.

<A1> *SV/MLS on the 3B4000 must maintain the same system call interface and operation as the uniprocessor version of SV/MLS.* From the onset, it was known that SV/MLS would require non-trivial changes for the port. This requirement was designed to localize the SV/MLS changes to the kernel components. Any kernel change that did not alter the system call interface from that of the uniprocessor SV/MLS would aid in reducing the number of changes to SV/MLS user-level commands and libraries.

<A2> *Performance degradation on the 3B4000 because of SV/MLS must be no more than 5% compared to a non-SV/MLS 3B4000 system as measured by industry accepted benchmarks.* This requirement is the same as the SV/MLS uniprocessor performance requirement. It is present to prevent the creation of a *secure brick*. Performance degradations greater than 5% will cause users and system administrators to disable the security features of SV/MLS and, therefore, make its presence on the system useless.

<A3> *The 3B4000 SV/MLS audit trail must be able to be written to any PE on the system.* An extension of a uniprocessor SV/MLS requirement, this is to assure that the audit trail can be written to any writable device no matter which PE that device is on.

<A4> *The 3B4000 SV/MLS audit trail must contain data to allow the determination of the PE(s) that a given event occurred on.* Since PE's are addressable objects in the 3B4000 UNIX System V, successful and failed accesses to them and the resources they control must be audited.

<A5> *A System Officer must be able to boot and shutdown APE's without shutting down the entire 3B4000 system. The booting and shutting down of APE's must be audited.* This stems from an original 3B4000 requirement. A goal of the 3B4000 is to provide long spans of uninterrupted service. As such, the shutdown of a given PE must be allowed without shutting down the rest of the system. Also, booting (shutting down) APE's adds (deletes) objects from the users' address spaces. As such, the booting (shutting down) of an APE must be audited.

The above requirements are listed to show the basis for design decisions described later which impacted various components of SV/MLS. As necessary, particular requirements will be called out at the relevant sections.

SECURE NETWORKING ISSUES ENCOUNTERED

When extending the concepts of secure computing to multiple CPU's, many concepts that are simple in a single processor system become somewhat more complex. Two issues that had to be dealt with in the SV/MLS 3B4000 port were distributed auditing and label management.

Distributed Auditing

On a single processor system, a single thread of control is guaranteed since there is only one CPU executing instructions and, therefore, creating auditable events. In a multi-processor system or network, each CPU is generating auditable events. Also, each CPU may have its own clock. This is especially likely in a network. As such, when auditing events on a multi-processor system or network, the time-stamp of an audit record must be associated with the clock of the CPU that generated it.

An audit record from a given CPU must be ordered with respect to the records of the other CPU's. Determining the skew between the clocks or synchronizing them becomes important if the audit data is to have meaning. Also, since multiple CPU's can cause *simultaneous* events, there must be a way to determine which events are actually caused by other events and which are totally unrelated.

In addition to having correct time-stamps and proper ordering of the audit data from multiple CPU's, secure multi-processor and networking products must decide where the audit data is to be stored. Audit data could be stored at the processor local to the events until review of the data is required. Or, all the audit data could be sent to a central collection point as the audit records are being generated. While centralized auditing provides ease of access to the whole networks audit data, it could increase network traffic to the point where the entire network grinds to a halt. Localized auditing keeps the network clear, but each processor is required to have a substantial amount of local storage for the audit data. The latter solution is unacceptable in a diskless workstation environment.

Label Management

Similar to the issue of distributed auditing, label management has to do with global access to a set of globally significant data. In auditing, this data is almost entirely *write-only*. Conversely, the data and operations associated with label management are almost entirely *read-only*. As in distributed auditing, centralized versus localized label repositories can be used. The two primary concerns are:

- network traffic associated with distributing labels around the network, and,
- synchronization of labeling information between label repositories.

A centralized label repository requires no synchronization, but all but one processor must do non-local I/O to get a label. On the other hand, localized label repositories eliminate most, but not all, network traffic associated with label passing. Some traffic must still occur to update the label repositories of processors not local to where a new label has been defined for the overall distributed system or network. In addition, a communications scheme must be developed to provide *strong* assurances that the label repositories are always synchronized for access decisions to be made correctly.

SV/MLS has a file that is used, among other things, to translate the privilege associated with every subject and object to a human-readable label. When dealing with a multi-processor system or network, however, a mechanism for managing labels between CPU's must be implemented. For most distributed systems or a network where all the hosts have come to agreement on a labeling standard, the mechanism could be as simple as a shared resource that is accessible by all processors. In that case, an implementation similar to that of SV/MLS could suffice. For large or heterogeneous networks, the label management mechanism might have to be complex with labeling domains and mapping functions to translate between domains. This might be required if only subsets of the systems on a network could come to agreement about what label convention to use.

When communicating labeled information out of one of these clusters of systems to another cluster, a label mapping algorithm would have to be used to translate the labels of one domain to another.

3B4000 SV/MLS IMPLEMENTATION

This section presents the changes implemented in SV/MLS to meet the requirements and achieve workable solutions to the network security issues presented above.

Audit Subsystem

In analyzing the 3B4000, it was discovered that the MP synchronizes all its APE's clocks at least once per second. As such, for the SV/MLS 3B4000 port, no additional work was required in the area of clock synchronization.

AT&T has already tackled the data ordering issue in its securing of UNIX Remote File Sharing (RFS) for use with SV/MLS.[6] The SV/MLS RFS strategy is to audit accesses on the *server* that contains the *object* being accessed. As the SV/MLS 3B4000 port progressed, other similarities between the 3B4000 distributed architecture and RFS were noted. As such, it was decided to use the AT&T RFS/MLS auditing strategy for the 3B4000. This became known as the "Object PE Records Audit," or OPERA rule. Given the single-threaded, non-preemptable nature of the 3B4000 PE kernels, and because auditing at the object PE assures that audit records relevant to the object remain ordered, implementing the OPERA rule in the SV/MLS 3B4000 audit subsystem was enough to meet the SV/MLS auditing requirements.

For placement of the audit data, it is possible for the three (3) audit subsystem components described earlier to be resident on up to three (3) different PE's. This capability was implemented to meet *<A3>*. However, for performance reasons, it was later found that all three components should reside on the same PE. The remainder of this paper assumes that all three components are on the same PE, called the Security Audit Trail Processing Element (SAT PE).

The SV/MLS 3B4000 port implements a centralized audit trail. The SAT PE is the central repository for audit data. This solution was chosen because some 3B4000 PE's do not support local mass storage. As such, these PE's would have to send their data to a remote audit trail repository anyway. Therefore, the centralized audit trail was chosen in the interests of a simple solution applicable to all PE's. All PE's send their audit data to the SAT PE via the A-bus. As such, only the SAT PE needs to allocate a storage area for audit trail data. However, A-bus bandwidth is consumed by audit data traveling to the SAT PE from all the other PE's.

The question arises, "What should the other PE's do when the SAT PE goes down?" In theory:

a. all subsequent reference monitor requests on all other PE's should fail/hang, and/or,

b. no more auditable events of any kind can occur on the entire system.

In determining whether reference monitor requests should fail or user processes should be suspended, it was found that it didn't matter. If the SAT PE goes down, it cannot be rebooted without tripping an audit point. Granting access to and executing the program to reboot a PE are auditable events.

Under normal circumstances (i.e., the SAT PE is up), the SAT_START records that audit the boot of a PE are cut by the MP and sent to the SAT PE. Under the SAT-PE-crashed case, the MP cannot audit the (SAT PE) boot event and will therefore fail or hang, depending on the implementation chosen. With this the whole 3B4000 will eventually suspend, since the MP runs the A-bus. Sooner or later each APE will request an MP service and hang waiting for the request to complete. In light of this, the "if there's no SAT PE, shutdown" solution was chosen. If a PE generates an auditable event and can't send it out over the A-bus to the SAT PE, the originating APE attempts to send a "go to single-user mode" request to the MP. If this request succeeds, the APE waits for the MP to shut down the whole system. If the request fails, the APE panics.

A final note on the auditing requirements. To meet *<A4>*, all 3B4000 SV/MLS audit trail records have fields that identify the PE's involved in the auditable event that generated the record. The PE numbers stored in the records depend on the auditable event type and the OPERA rule. To meet *<A5>*, a new audit trail record, SAT_STOP, was implemented for the auditing of the crash or intentional shut-down of an APE. Also, the meaning of the SAT_START record was expanded to include the booting of APE's in addition to the booting of the MP.

Choosing a SAT PE

The choice of SAT PE can make a difference in the performance of the whole 3B4000 SV/MLS system. This section provides guidelines determined through review of the 3B4000 design documents and by experimentation on the live development and production systems.

The first thing determined was that, if possible, the SAT PE should not be the MP. If any APE has mass storage capabilities, it should be used as the SAT PE, over the MP. The MP is best left free to move traffic around the A-bus.

Given the OPERA rule, the SAT PE should be the one that *contains the greatest number of objects accessed most frequently.* This set of objects includes files, directories, pipes, and System V IPC structures. The placement of the audit trail files local to the bulk of the auditable activity greatly reduces the amount of A-bus traffic, offloading the MP and generally improving throughput.

As a final note on SAT PE placement, it was found that making the SAT PE and the PE that contains **/bin**, **/usr/bin**, and **/tmp** one and the same helps throughput significantly. Actions on objects in these three file-systems can amount to the bulk of the auditable events on a UNIX system.

Label Management Subsystem

The uniprocessor SV/MLS maintains all label information in use on the system in the labels file. This was not changed for the SV/MLS 3B4000 port. There were concerns at first that such a file on only one PE would greatly increase A-bus traffic, causing the product not to meet <*A2*>. But, by using per-PE caches similar to the cache of the uniprocessor product, it was believed that <*A2*> could be met. A single file with label information also eliminates the need to synchronize many such files on multiple PE's. The MP was chosen as the keeper of the labels file for a variety of reasons. The most significant reason was that since the MP is always the first PE to be booted and it must have access to the labels file to boot itself and the other PE's, then, if there is only one labels file, it must be on the MP.

Since label information is centrally stored on the MP, there must also be a way for APE's to get labels. The algorithm used is a convenient extension of the uniprocessor SV/MLS version. In the uniprocessor version, if the access control software requires a label, it first looks in the labels cache. If the label is not there, one or more reads of the labels file are performed. When the label is retrieved, it is placed in the cache for future reference. If the cache is full, the least used label (determined by a per-cache-entry hit count) is overwritten by the newly gotten one.

On the 3B4000, the same algorithm is used on the MP, since the labels file is local. However, if an APE's access control software requires a label, it first checks an APE-local cache. If the label is not found, the APE sends an A-bus message to the MP requesting it. The MP first checks its cache and if the label is not there, the MP gets the label from the labels file.

The MP cache check is important because the odds are that the label is already there from a previous access check. When users log in, among the first things they access is **/etc/profile**, which is on the root file-system of the MP. By virtue of the OPERA rule, the access checks are performed on the MP and the label is deposited in the MP labels cache. Even if a user never accesses another MP object, the label remains in the cache (within the constraints of the replacement algorithm). This saves a considerable amount of disk accesses which are much more expensive than A-bus messages.

Any distributed or network label management system must account for the deletion of labels. When a label is deleted, instances of the label in caches throughout the 3B4000 system must be rendered invalid. The SV/MLS 3B4000 port uses a global reset message which is sent by the MP to all APE's when a label is deleted. This message causes each APE to invalidate *all* entries in its cache. The caches must then be reloaded over time as labels are required. This is not the most elegant of techniques. But, given that label deletion is a relatively rare event, the technique was acceptable.

System V/MLS Commands and Libraries

Because of <*A1*>, few user level routines had to change. Those that did change were augmented to handle additional capabilities and/or features of the 3B4000.

The most evident difference in the SV/MLS user interface on the 3B4000 versus a uniprocessor system is the *prlbl*(1S) command. The interface to and output of `prlbl -s` had to be changed to accommodate multiple labels caches (one per PE). An administrator can use `prlbl -s` to display label cache hit statistics. Based on this information, the administrator can "tune" the cache for optimal performance. Since each PE has its own labels cache, the interface used by *prlbl*(1S) to get the statistics from the SV/MLS kernel modules had to be changed. Also, the display of the cache information was changed. The old display is shown in (**Figure 2**).

```
# /mls/bin/prlbl -s
cache hits = 2638826
one read from disk = 52
more than one read from disk = 9
cross product found on disk = 4356
```

Figure 2. Output of a uniprocessor `prlbl -s`

For the 3B4000, the *prlbl*(1S) display has been augmented to show PE specific cache hit information. The new display is shown in (**Figure 3**).

```
# /mls/bin/prlbl -s

        | PE |   chit   | dhit1    | dhit2    |   rhit   | conlbl   |
        +----+----------+----------+----------+----------+----------+
        | 121|  10482652|     14970|     20347|          |     10788|
        |   0|      1010|          |          |        16|         0|
        |  80|   2918229|          |          |     16785|      6076|
        | 120|     31919|          |          |      3781|         0|
```

Figure 3. Output of a 3B4000 `prlbl -s`

The columns specified are:

chit	- cache hits,
dhit1	- retrieved label from disk with one read,
dhit2	- retrieved label from disk, required more than one read,
rhit	- retrieved label via A-bus transaction,
conlbl	- *cross-product* privilege constructed from group of one privilege and label of another.

ACHIEVEMENT OF REQUIREMENTS

The previous sections have shown the implementation choices made to meet the requirements and network security issues of the SV/MLS 3B4000 port. This section presents the steps taken to assure that these implementation choices were valid.

Among the best ways to determine if *<A1>* was met was to examine the amount of change to the system test software/procedures. AT&T offers the *AT&T System V/MLS Security Test Package,*[7] (STP) which is intended to determine whether a particular implementation of SV/MLS meets the SV/MLS security requirements and interface definition. This package is primarily for source customers to evaluate a port of SV/MLS to their particular platform(s). It was expected that STP would have to be changed to test the PE fields in the audit data, plus the new SAT_STOP and enhanced SAT_START records. No other changes were expected before starting the testing of the port. After testing was completed, the only changes, except for those expected, were to accommodate the 3B4000 package installation mechanisms, device names, and directory hierarchy, which differ slightly from those of the AT&T 3B2 or 6386, the platforms STP was originally designed for.

Early on in the port, the cache hit statistics showed that *<A2>* would be met. From the data in (**Figure 2**), the uniprocessor cache hit ratio comes out to show that 99.8 percent of all label references are resolved with cache hits. Using the data from (**Figure 3**), the average 3B4000 SV/MLS labels cache hit ratio was determined to be 99.5 percent.[2]

As the port neared completion, the Neal Nelson benchmark was used to compare the 3B4000 SV/MLS performance figures against *vanilla* 3B4000 UNIX System V figures, taken on the same system, before the port began. The performance degradation was not measurable using the Neal Nelson benchmark, beyond differences attributed to statistical error. This showed that *<A2>* had been met.

In addition to data provided by the Neal Nelson benchmark and the Security Test Package, the SV/MLS 3B4000 port currently has over 6500 system-hours of hands-on use. Since the final load, no SV/MLS related shut-downs have taken place. Therefore, a good deal of confidence exists that the implementation is sound.

LIMITATIONS OF 3B4000 PORT RESULTS

Although the SV/MLS 3B4000 port provided many insights into network security issues, two shortcomings in the solutions were found. These two shortcomings are presented in this section.

Clock Synchronization

The 3B4000's existing clock synchronization simplified the implementation of the audit trail considerably. Had the synchronization not been there, it would have had to have been implemented. This surely would have made things much more complex. Most networks, however, *do not* have synchronized clocks between the hosts on the network. As such, for a usable audit trail to be created based on the SV/MLS design and the 3B4000 porting work done to date, some type of synchronization mechanism must be developed. Given that, the lessons learned from the 3B4000 port can be applied to the remainder of the network audit trail implementation.

Limitation of the OPERA Rule

The OPERA rule has one shortcoming. Although the OPERA rule implementation allows system administrators to answer questions like:

— Who was the last person to open /etc/passwd for writing?

— What happened first; did Lucy write to the file, xyz or did Andy read it?

Where the OPERA falls short is in answering questions like:

— Did Audrey write to the file, ABC on PE80 first, or did she read the file, QXR on PE121, first?

The OPERA rule cannot be used to answer this question. And, if the two events mentioned above are close in time (within the granularity of one system clock tick), the ordering cannot be accurately determined in the

2. Both the 3B4000 and uniprocessor data come from configurations with 10 entries per labels cache.

3B4000 SV/MLS audit trail. An analogous rule, "Subject PE Audits Request," or SPEAR, could answer a question such as the previous one. But, given the same timing conditions, SPEAR could not answer the questions that OPERA could. A combination of OPERA and SPEAR must be implemented to completely order the audit trail for all cases. Because of other facets of the 3B4000 architecture beyond the scope of this paper, SPEAR was not implemented, as the events that it is needed to distinguish are rare.

APPLICATIONS TO SECURE NETWORKING

Building on the experiences of the SV/MLS 3B4000 port, one could apply the results of the port to produce an MLS network based on the SV/MLS design. This section applies the SV/MLS 3B4000 port design decisions to the secure networking issues, distributed auditing and label management, described earlier.

Distributed Auditing

The Neal Nelson benchmarks and day-to-day use of the SV/MLS 3B4000 system show that the implementation of centralized auditing and the associated network traffic for audit data necessary to achieve a B1 evaluatable network is possible without sacrificing performance. The centralized approach also provides a solution for networks that have components, such as diskless workstations, that don't have the ability to store audit data locally.

The effort to determine which PE of a 3B4000 should be the target for the centralized audit files showed that the choice of a target for audit trail data does make a difference in performance. Given the OPERA rule, the audit trail should be collected *on or as near as possible to the network component(s) that contain the greatest number of objects that are accessed most frequently.* The goal is to limit network audit trail messages as much as possible. The SPEAR rule mentioned earlier, adds the additional constraint that auditing data should be collected *on or as near as possible to the network component(s) that contain the greatest number of* subjects *that produce the most audit data.* OPERA and SPEAR need not be conflicting rules. In client/server networks with server processors containing database objects and server process subjects, OPERA and SPEAR work together well. The database accesses account for a large amount of the auditing affected by OPERA and the actions of the servers account for a large amount of the auditing affected by SPEAR. Since both types of auditing come from the same network component, the audit files could be centralized on or near the most active server(s), thereby satisfying both OPERA and SPEAR. Of course, real-world solutions are never so cut-and-dry. But, the use of OPERA and SPEAR as guidelines, if not rules, can help determine where to place auditing in a secure network.

Label Management

Like distributed auditing, the performance data and system usage experience show that centralization of label information is practical from a performance perspective. With proper use of caches, the network traffic associated with distributing labels from a central repository is not prohibitive. For distributed systems and networks with all elements in agreement on labeling, efficient centralized label management proves to be realizable. For those (larger) networks mentioned earlier that cannot come to agreement, a central label repository per *agreeable* cluster is also shown to be realizable by the SV/MLS 3B4000 results.

A subtle, but important concept that emerged from the label distribution algorithm was the checking of a labels cache local to the label repository before going to the repository itself. If a user is likely to access an object on the network component containing the centralized label repository, then that component's labels cache will contain the label in question when the access control software of another component requires it. One real-world occurrence of this phenomena is a network where a centralized database is searched to determine if a user has access to the given network. If that database is on the network component that also contains the label repository, then the cache searching technique presented above will decrease the number of times the repository (rather than the cache) is searched for labels. Such a configuration would exist in a network architecture with a central *security server.*

CONCLUSIONS

This paper described the port of AT&T's System V/MLS to the AT&T 3B4000 super-minicomputer. During this port, the 3B4000 distributed architecture's resemblance to a network unexpectedly provided answers to and performance data on MLS networking issues that were not part of the original goals of the porting project. The SV/MLS 3B4000 port acted as a worked example to show that practical and efficient MLS networks can be built and that the SV/MLS design is a practical base for an MLS network. With the increasing demand for secure networking products and services, this provides welcome evidence that secure systems don't have to be unusable systems.

ACKNOWLEDGEMENTS

I thank Bill Leighton and Pete Dinsmore for letting me "find" the time to prepare this paper, as well as providing the funding and support. I thank Glen Gordon for sanity-checking potential designs and helping me understand the internal operation of the 3B4000 system. I thank Cheri Dowell for her work in system testing the SV/MLS 3B4000 port. I also thank the Department 46291 system administration staff for letting the port loose in their lab. And, finally, I thank all the members of Department 46291. By acting as an (involuntary) Beta site, they helped iron out the last kinks in the system.

REFERENCES

1. Bell, D. E., LaPadula, L. J., *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR-2997 Rev. 1, MITRE Corp., Bedford, Mass., March 1976.
2. Fish, R. W. (ed.), et al., *DSB-720300CD - AT&T 3B4000 Kernel, Component Design Specification, Issue 3*, AT&T Information Systems, May 20, 1988.
3. National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985.
4. *System V/MLS Multilevel Security (MLS) Module - Requirements, Design, and Implementation*, AT&T Bell Laboratories, August 9, 1990.
5. *The System V/MLS Security Audit Trail - Requirements, Design, and Implementation*, AT&T Bell Laboratories, June 27, 1990.
6. Johnson, E. M., *Secure Remote File Sharing on SV/MLS*, AT&T Bell Laboratories, June, 1, 1989.
7. *AT&T System V/MLS Security Test Package User's Guide*, AT&T Bell Laboratories.

AN EXPERT SYSTEM APPLICATION FOR NETWORK INTRUSION DETECTION*

Kathleen A. Jackson, David H. DuBois, Cathy A. Stallings

Computer Network Engineering Group, MS B255
Computing and Communications Division
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

This paper describes the design of a prototype intrusion detection system for the Los Alamos National Laboratory's Integrated Computing Network (ICN). The Network Anomaly Detection and Intrusion Reporter (NADIR) differs in one respect from most intrusion detection systems. It tries to address the intrusion detection problem on a network, as opposed to a single operating system. NADIR design intent was to copy and improve the audit record review activities normally done by security auditors[1]. We wished to replace the manual review of audit logs with a near realtime[2] expert system. NADIR compares network activity, as summarized in user profiles, against expert rules that define security policy, improper or suspicious behavior, and normal user activity. When it detects deviant (anomalous) behavior, NADIR alerts operators in near realtime, and provides tools to aid in the investigation of the anomalous event.

1 Introduction

The authentication and access control system in any network is the first defense against intruders from outside. At Los Alamos, we define authentication as the identification of a user with reasonable assurance that the user is who he or she claims to be. Access control is defined as a mechanism of restricting access by authenticated users to those parts of the network consistent with their clearance and need-to-know. It is clear, given the industry-wide frequency of break-ins by outsiders, that authentication and access control mechanisms can be compromised or bypassed. They alone cannot supply assurance against penetration by outsiders. Also, outside "hackers" are not the only source of security problems. Far more often they are a result of abuse by the privileged insider. Even the most secure system is vulnerable to abuse by insiders who misuse or try to misuse their privilege. This is obvious from well publicized reports of incidences of unauthorized access and removal of classified information by insiders from otherwise secure computer systems.

In a large, complex, and rapidly changing computer network such as the ICN it is not realistic to expect to identify all security loopholes and vulnerabilities. Even if identified, it is not a given that they can be closed, since it may be impossible or impractical to do so. A primary reason for this is the need to strike a balance between security and the provision of convenient services to network users. Given the acknowledged doubt in the completeness of current security measures, we are tasked to identify and implement new technologies that support network security.

An auxiliary line of defense against both intrusions by outsiders and insider misuse is the maintenance and review of an audit record of important network activity. In our case, maintenance of an adequate audit record presents few problems. This has been a required activity at Los Alamos for many years. However, attempts at audit record review result in security auditors wading through huge quantities of output in an ineffective attempt to spot invalid activity. The sheer vol-

*The Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36. This work was performed under auspices of the United States Department of Energy.

[1] Los Alamos security auditors are specialists whose responsibility is to ensure the security of the ICN. They include security officers such as the CSSO and CSSM, and their staffs.

[2] For our purposes, we define a near realtime application as one that responds to data or user input in one to 30 seconds.

ume of data makes it nearly impossible to detect suspicious activity that does not conform to a few obvious intrusion or misuse scenarios. Even these may be missed. To make audit review effective, the auditors need the capability for automated analysis of the audit record. This capability combines the knowledge of security experts with a computer's capability to process and correlate large quantities of data. When done in near realtime, the auditors can be notified of suspicious activity quickly, and direct action taken to trace and stop an identified penetration attempt or other misuse. This is the essence of an intrusion detection system.

2 Target System

The Integrated Computing Network (ICN) is Los Alamos National Laboratory's main computer network. It includes host computers, file storage devices, network services, local and remote terminals, and data communication interfaces. The core of the ICN includes the main host supercomputers and their support devices. Through the ICN, any user inside the Laboratory may access any host computer (with authorization to do so and use of an approved access path) from office workstations or terminals. Outside users typically access the ICN through telephone modems, leased lines, or one of multiple world-wide networks. The core ICN has more than 8,000 validated users.

The ICN consists of a unique arrangement of four "partitions," in which resources are dedicated to specific levels of processing. Each partition limits access to only those users cleared for the most sensitive information processed in the partition. A system of dedicated, special function, ICN nodes enforce partitioning throughout the network. These *service nodes* perform specific services in the ICN, such as user authentication, access control, job scheduling, file access and storage, file movement between partitions, and hardcopy output. They are physically protected, have tightly restricted access, run only that software needed to perform a specific service, and do not execute user programs. Only these dedicated nodes may service multiple ICN partitions. Each of these nodes must produce and maintain an audit record of its activity. They are the ICN systems targeted for our intrusion detection effort.

3 Overview

Until recently, security auditors manually reviewed ICN audit records to identify potential security violations. Given the size of the audit records, manual review was limited to a small sampling or a cursory scanning. The auditors found many security violations, but there was no way to evaluate the general success or completeness of their effort. Also, the Laboratory's Internal Security (ISEC) office often requests audits that cover weeks of audit data from months or years in the past. As there was no automated way to do these audits, considerable effort was expended in completing them. It was for these reasons that development of an automatic audit record analysis, or intrusion detection, system was undertaken at Los Alamos.

The early research of Dorothy Denning and her colleagues, and the IDES research and development at SRI International, has heavily influenced intrusion detection development at Los Alamos. Denning proposed monitoring standard operations on a target system for deviations in usage. Her early research tried to define the activities and statistical measures best suited to do this [1, 3], and continued with the development of an IDES prototype [4]. Teresa Lunt and her colleagues continue this research with the development of the IDES system [5, 6, 9, 13]. They expanded the original concept by adding an expert system component that addresses known or suspected security flaws in the target system. IDES research has served to demonstrate two things. First, that statistical analysis of computer system activities provides a characterization of "normal" system and user behavior, and that activity deviating beyond normal bounds is detectable. Second, that known intrusion scenarios, exploitation of known system vulnerabilities, and violations of a system's security policy are detectable through use of an expert system rule base. The IDES approach puts a primary emphasis on the statistical detection of deviations from normal user and system behavior. The expert system is intended to catch those invalid activities missed by the first means [10].

Several intrusion detection systems have in recent years adapted the Denning model to their particular problem [7, 8, 11]. However, where the Denning model and most intrusion detection systems target specific operating systems, our effort addresses a *network* connecting many host systems, but not the hosts themselves [15]. Where Denning addressed the standard operations on a specific operating system (system logon, program execution, file and device access) we wished to address the standard operations on our network. The problems are similar in many respects, but with some important differences. While the ICN contains many standard functions such as those found on an operating system (authentication, access control, file access and storage, job control), these functions are distributed across the network. Also, the ICN implements a *distributed* multilevel secure system (the system of partitions and the controls over them), that must be monitored closely by any intrusion detection system. Nonetheless, if we view the ICN as one large distributed operating system, then the Denning model applies well to the problem of network intrusion detection.

Current network intrusion detection efforts have taken one of two approaches. One approach is to target network traffic at the service and protocol levels [12]. The second approach collects data from separate hosts on a network, for processing by a centralized intrusion detection system [14]. Although NADIR does not capture network traffic, it targets service level activity by targeting the service nodes that handle and log standard ICN service operations. We decided to target the *service nodes* because of their critical nature, to keep the quantity of data to be processed at a manageable level, and because their audit record is sufficient to support an effective intrusion detection system.

4 Working Prototype

Once we decided to apply intrusion detection to the ICN service nodes, we adopted three basic technical goals. These goals support development of a flexible system that we could expand to multiple target systems. The first goal was to limit the audit record to that currently supplied by the target systems. The second, to keep target system changes to a minimum. The third, to avoid degradation of target system performance.

Because the ICN is a large, long-established network that has changed constantly over the last fifteen or so years, we had to take the following peculiarities into account:

- The Los Alamos developed network protocols are non-standard, so are not compatible with off-the-shelf software.
- The ICN service nodes comprise several different hardware configurations, that run a variety of operating systems.
- The software on most service nodes has been subject to many changes and upgrades, and is programmed in several different languages.
- While each service node must maintain an audit record of its activity, the format and content of the audited data differ greatly from system to system.

To support expansion to these various multiple target systems, we made three design choices. First, to use dedicated workstations for intrusion detection processing. Second, to use flexible off-the-shelf interface and database software, that supports data translation between different operating systems and enables the merging of data into a single extended database. Finally, to limit required target system changes to the capability to collect the proper audit record of user activity, transform the data into a specified canonical format, and transmit it to NADIR. Also, we designed NADIR software in a modular fashion, so that new target system expansions can be handled with a minimum of effort.

NADIR is to be implemented on a set of dedicated workstations, each of which will receive and correlate data from multiple target systems. As we add more target systems to NADIR, we plan a network of workstations, each contributing to a distributed database. This approach minimizes the impact on target system performance, enables the collection of data from multiple diverse systems, and provides for maximum security. Ethernets will connect the workstations to the target systems and to each other, and we will implement a standard network protocol.

The NADIR prototype consists of one workstation, a SUN SPARCstation[3] with two 327 MByte disks. It uses the Sybase[4] relational database management system and a Los Alamos designed expert system. Sybase provides tools used to structure, maintain, and display all data on the system. The expert system is programmed almost entirely in Transact-SQL, an enhanced version of the SQL database language supplied by Sybase. Transact-SQL provides such capabilities as stored procedures, triggers, system administrator tools, and control flow language features, used extensively in NADIR. NADIR communicates with each target system over a dedicated secure ethernet link.

The prototype NADIR currently monitors Network Security Controller (NSC)[5], Security Assurance Machine (SAM)[6], and Common File System (CFS)[7] activity on the ICN. The NSC is a DEC-8250[8] machine, which runs the VMS operating system. The SAM is a DEC-730 machine, which runs the UNIX[9] operating system. The CFS is a IBM 3090 mainframe. NSC and SAM data is transmitted directly to NADIR, while CFS data is passed to an intermediate VAX/VMS system before transmission to NADIR. The changes called for on each target system were minimal. Communication with NADIR by a target system calls for only the installation of Sybase supplied interface software, and the use of a standard DECnet or TCP/IP protocol. DB-Library packages for Fortran and C provide the interface to Sybase. The Multinet[10] software package provides an implementation of TCP/IP under VMS. We changed each target system code as little as possible. The target system must only format the audit record for NADIR and transmit it immediately after its occurrence. NADIR required data processing has not resulted in any measurable degradation in system performance on any target system.

5 System Design

We are applying NADIR to the ICN service nodes in a sequence of planned phases. Each phase includes analyzing a node individually, processing its data separately, then integrating it into the NADIR system. As we add new nodes to NADIR, we correlate their user activity record with earlier included nodes to produce more complete profiles of ICN activity. Eventually, this will allow the tracking of users from the time they enter the ICN, until they leave the network. With the addition of each node, we define new expert rules that use the expanded information available. The rules describe more elaborate scenarios of invalid or suspicious user activity, and will, over time, improve the discrimination and judgement of the system. We have integrated the NSC, the SAM, and the CFS into NADIR. Work is in progress to integrate the Facility for Operator Control and User Statistics (FOCUS)[11] and the Print and Graphics Express Station (PAGES)[12]. These are all the nodes initially targeted for prototype development.

[3] SUN SPARCstation and SUN workstation are trademarks of SUN Microsystems, Inc.
[4] Sybase, Transact-SQL, and DB-Library are trademarks of Sybase Corporation.
[5] The NSC is a dedicated, single-function computer through which all ICN user authentications must pass.
[6] The SAM controls and audits the down-partitioning of unclassified files between partitions in the Common File System (CFS).
[7] The CFS is a large, centralized file management and storage system that provides long-term file storage in all ICN partitions for ICN users.
[8] DECnet, VMS, DEC-8250, and DEC-730 are trademarks of Digital Equipment Corporation.
[9] UNIX is a trademark AT&T Bell Laboratories.
[10] Multinet is a trademark of TGV, Inc.
[11] FOCUS provides operations control, batch job scheduling, and accounting control for the ICN.
[12] PAGES produces listings, graphics, and formatted document output for ICN users. Output is subject to partition and classification restrictions.

The NADIR system has six functional components; Data Collection, Data Processing, Anomaly Detection, Report Generation, Event Assessment, and the User Interface. Figure 1 illustrates their relationship to each other.

5.1 Data Collection

NADIR monitors target system activity as it happens. Each audit record describes a single event. Audit records from different target systems vary in format and contain mostly unique data, a result of the functionally different tasks done by those systems. Whatever the system, the audit record will contain a unique ID for the ICN user, the date and time of the user's activity, fields that describe the activity, and any errors that might have occurred.

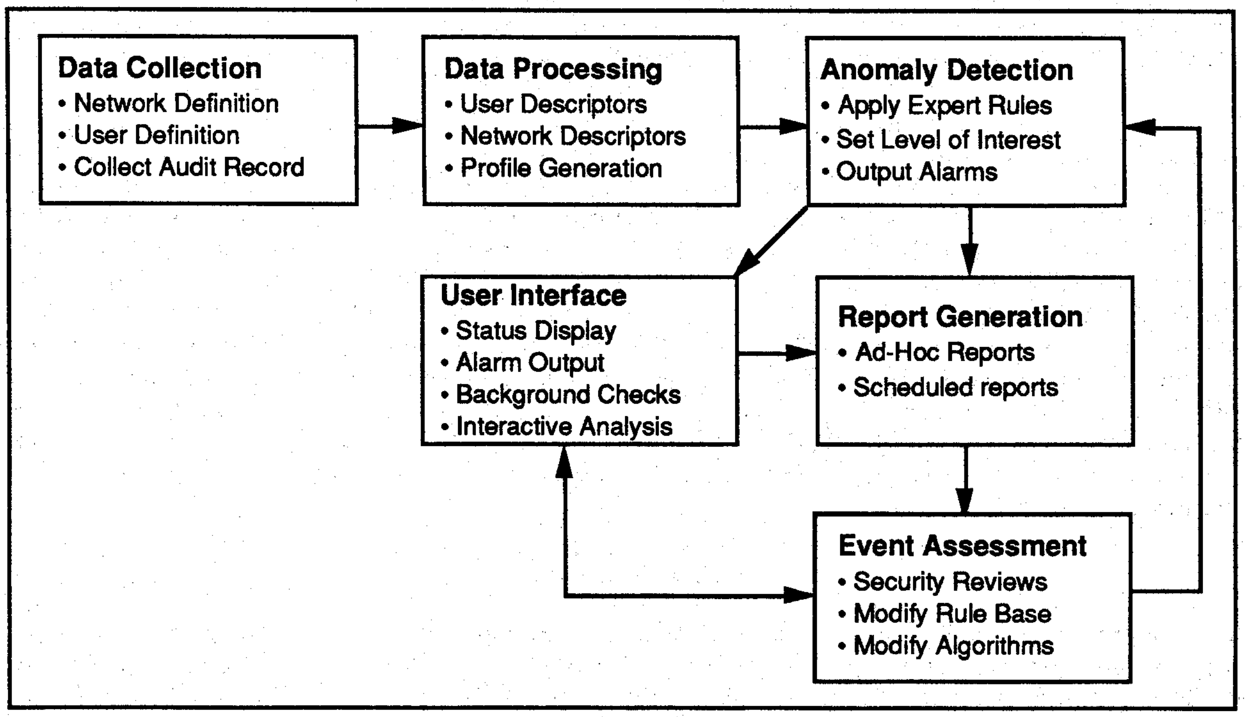


Figure 1: NADIR System Model

5.2 Data Processing

NADIR summarizes all user and system activities, as represented by audit records from the target systems, into statistical profiles. These profiles are a description of current behavior in a set of defined parameters. NADIR maintains profiles for both separate ICN users and for a composite or total of all ICN users. They contain measures (count statistics) that summarize user activity. These measures keep a record of the occurrences of a particular event during a specified time. NADIR updates the profiles when it receives an audit record. It parses the data from each audit record and increments the proper profile measures. NADIR maintains past profiles for comparison purposes and as a permanent record.

5.3 Anomaly Detection

NADIR finds events in either the contents of a single input audit record or from an examination of the user profiles. Single audit records define an event when any of the data fields in the record match a specified pattern. Events detected in the profiles represent activity that is spread across multiple audit records. They define an event when the profile measures match a specified pattern.

NADIR compares proper and expected activity to observed events within either the audit record or the profiles. It does this through the application of expert rules, and identifies deviations[13]. NADIR assigns each deviant event (or anomaly) a Level-of-Interest[14]. It bases the Level-of-Interest on the number and type of rule that the user's behavior has fired. NADIR applies the Level-of-Interest to each unique user, host system, or entry point into the network. Every fired rule increases the Level-of-Interest, though the firing of one critical rule may be enough to bring immediate attention to the event. The current security status for each user and system is provided in the combination of Level-of-Interest and record of fired events.

5.4 Report Generation

NADIR generates anomaly reports from deviant events. The frequency of reports is dependent on the Level-of-Interest associated with each event. All events are documented in routine weekly reports. Those events determined to be very interesting, but not critical, are output in daily reports. Very suspicious events of a critical nature, such as a probable attack under way, are output immediately. NADIR generates detailed follow-up reports as part of any investigation.

5.5 Event Assessment

Upon receipt of a NADIR report, whether critical or routine, security auditors review all anomalous activity. To process anomaly reports quickly, specific auditors investigate certain categories or types of ICN users. They review each anomalous user in detail, and decide whether to investigate further. This may include interviewing the user. If the user's activity warrants it, the user is blacklisted[15] during the investigation. The auditors file a short report at the completion of each investigation, giving details of its resolution. They supply this information to us, so we may have immediate feedback on system performance. The auditors hold periodic reviews to evaluate NADIR effectiveness and to make recommendations for improvements. We use their feedback to change the expert rules on NADIR and improve the discrimination and judgement of the system.

5.6 User Interface

The user interface uses Sybase front end tools, graphics packages, and Los Alamos designed routines to provide a preliminary interface for the knowledgeable user. It provides warnings, alarms, and status displays. For users who have the proper access and privilege, the user interface allows a choice of built-in or ad-hoc queries against the raw audit data, the separate user and composite profiles, and status information. Data may be displayed in a variety of ways, including graphically, and reports generated. In addition, NADIR provides tools for interactive background analysis of current and past activity. It maintains indefinitely the audit data needed for this activity.

6 Expert Rules

An expert rule base has separate reasoning rules encoded in a condition-action form (if-then-else statements in the old days), that provide the criteria for end determination. The rules watch for unusual separate events and attempt to evaluate the meaning of a group or series of events. NADIR expert rules, whether they are rules that enforce security policy or result from a statistical determination of normal behavior, define an expected standard of behavior for all users.

[13] The identification of a deviation by an expert rule is generally referred to as having "fired" or "triggered" the rule.

[14] The Level-of-Interest is the calculated seriousness of an event.

[15] A blacklisted user is denied access to the ICN by the NSC. Removal of the blacklist requires the prior approval of security personnel.

The NADIR rule base includes four logical filters; each designed to separate out certain types or levels of anomalous activities. Following a knowledge engineering approach successfully implemented at Textronic [2], the rule base definition started with the abstraction of the well-understood part of the problem. This included ICN security policy and well-defined invalid and suspicious behavior, which resulted in rules for the Characteristic Filter. Report requirements supplied rules for the Report Filter. From there evolved further refinements, implemented in the Misuse and Attack Filters. These rules involve heuristic associations that sometimes make intuitive leaps not always explicitly justified. NADIR activates the rule base filters in stages, as illustrated in Figure 2.

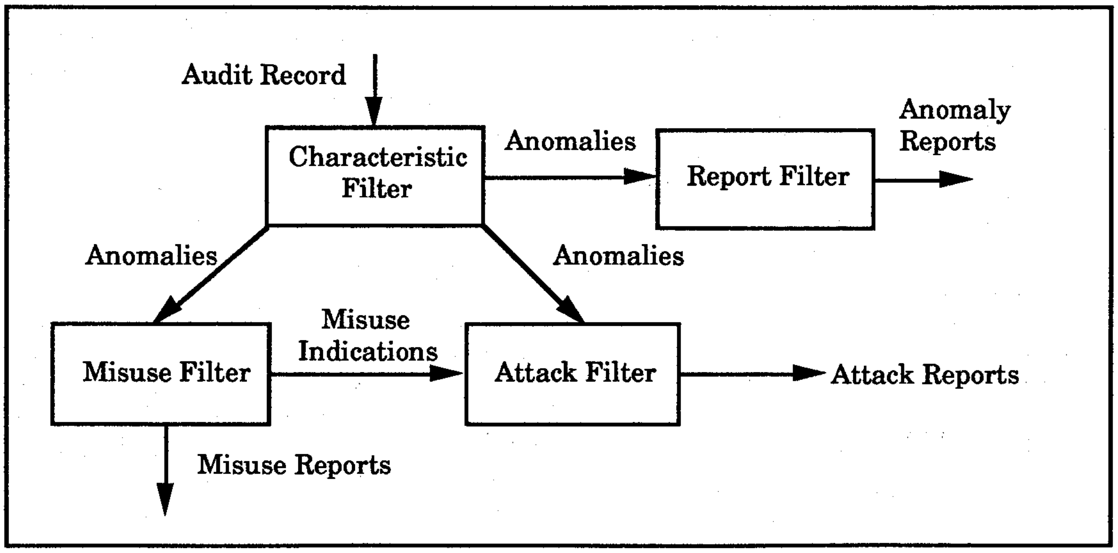


Figure 2: NADIR Rule Base Structure

• *Characteristic Filter* - applies rules that are straightforward descriptions of simple activities; each serving to distinguish a separate feature of anomalous behavior. NADIR applies these rules individually; it does not correlate one with another. It assigns a Level-of-Interest to each anomaly defined by these rules. This Level-of-Interest, as applied to each user or system, is incremental; with each rule fired it increases by a specified amount.

• *Report Filter* - applies rules to the anomalies output by the Characteristic Filter, to produce appropriate reports of anomalous behavior.

• *Misuse Filter* - applies rules to the anomalies identified by the Characteristic Filter. These rules try to identify patterns of anomalous activity that have a good chance of being systematic misuse. They specify what action to take when fired, such as the output of warning messages.

• *Attack Filter* - applies rules that try to correlate the recorded Characteristic anomalies and Misuse Indications with various Attack Scenarios. Attack Scenarios identify patterns of anomalous activity that have a good chance of being attacks on the system. They specify what action to take when fired, such as the output of alarm messages.

6.1 Characteristic Rules

NADIR applies Characteristic rules to either the input audit record or to profile data. As it finds each anomaly, it either generates or updates the Anomaly Record, whichever is appropriate. The

Anomaly Record includes a Level-of-Interest for the involved user or system, and an indication of the fired rule. Characteristic rules fall into three basic categories:

1. Security Policy - These rules are the implementation of ICN security policy. They result from interviews with security personnel and documentation reviews. They detect and immediately report potential or certain security violations. An example of a security violation rule:

```
IF NADIR has detected an "Improper Location" error,
   AND the terminal used is in the Open Partition,
   AND the password used is classified,
THEN update the Anomaly Record; assign the user a high Level-of-Interest.
EXPLANATION: Use of a classified password from an unprotected terminal is
reason enough to consider the password compromised. The password will be
immediately invalidated.
```

2. Individual Anomaly - NADIR applies these rules to separate user profiles, to detect when a user's behavior departs from that which is normal and valid ICN user behavior. They result from statistical analysis of the past behavior of ICN users, and interviews with security personnel. An example of an individual anomaly rule:

```
IF the Failure Ratio[16] of a user is >n1,
   AND the user has logged on >n2 and ≤n3 times,
THEN update the Anomaly Record; assign the user a Level-of-Interest.
EXPLANATION: If a user has logged onto the ICN at least n2 times then the
user is not new to the ICN. Since the average ICN user has a Failure Ratio
that is much less than n1, then a Failure Ratio of n1 is significant. NADIR
applies a sliding scale of concern, balanced between the total number of
logons and the Failure Ratio, to this rule.
```

3. Composite Anomaly - NADIR applies these rules to composite user profiles, to detect when that activity departs from that which is normal and valid for the system. They result from statistical analysis of the past behavior of the composite of ICN users. An example of a composite anomaly rule:

```
   IF "Unknown User" errors are >n3/hour, OR >n4/day, OR >n5/week,
THEN update the Anomaly Record; assign the system a Level-of-Interest.
EXPLANATION: The normal number of attempted authentications that contain a
user number that is not valid for the ICN is statistically very consistent.
Extreme variations from this expected activity could be a sign of a break-
in attempt. NADIR applies a sliding scale of concern to this rule, that de-
pends on the variation from normal.
```

6.2 Report Rules

These rules do periodic checks of anomalous user activity levels, and define what reports to generate after specific intervals. Designated report intervals may be daily, weekly, or any other period. They analyze the Anomaly Record for the indicated interval, and generate reports that summarize and detail anomalous activity.

6.3 Misuse Indication Rules

NADIR fires these rules when it receives a sequence or combination of Characteristic anomalies that have a low chance of happening. They suggest possible serious misuse of the network. They do

[16] $\text{Failure Ratio} = \frac{\text{Invalid\_Logons}}{\text{Successful\_Logons}+\text{Invalid\_Logons}}$

not try to define anything as specific as an attack, but their firing shows something is seriously amiss. The following simplified Misuse Indication rule examines overall ICN user activity:

```
IF the Level-of-Interest for >n6 ICN users is >0,
   OR the Level-of-Interest for >n7 ICN users is >x,
   OR the Level-of-Interest for >n8 ICN users is >x + x/2,
   OR the Level-of-Interest for >n9 ICN users is >2x,
THEN output an immediate report, that includes an urgent warning message to
the user interface.
EXPLANATION: The number of ICN users who reach a particular Level-of-Inter-
est is statistically very consistent. Extreme variations from the normal
level of anomalous activity could be a sign of some type of organized mis-
use of the network. NADIR applies a sliding scale of concern to this rule,
that depends on the users involved and their Level-of-Interest.
```

The following simplified Misuse Indication rule examines the Anomaly Record of a separate user:

```
IF Characteristic rule 003 is set,
(a separate user has many logons this week)
   AND Characteristic rule 056 is set,
   (the user has an unusual distribution of logon tries during the swing
   and weekend shifts).
   AND Characteristic rule 053 is set,
   (the user has only unsuccessful ICN logon tries during the night shift).
   AND Characteristic rule 043 is set,
   (the user has an unusual distribution of unsuccessful logon tries on the
   weekend).
   AND Characteristic rules 040, 041, 044, 045, 046, and 047 are not set,
   (the user does not show a like pattern of failures during the day shift
   or on weekdays).
THEN output an immediate report, that includes a message to the user inter-
face.
EXPLANATION: The fired Characteristic rules show a greater than normal
usage of the ICN, combined with abnormal usage during off hours. Also, the
user has had an abnormal number of failures during off hours while not
showing a like pattern of failure during normal working hours. This could
be a try at masquerading, and is surely suspicious.
```

6.4 Attack Scenario Rules

These rules may define one Characteristic anomaly or Misuse Indication, or a combination of these, that have a low chance of happening. They suggest a known or postulated attack. It is the sequence and combination of these rules that make for an increasing certainty that an attack may be proceeding. Attacks are events that could lead to the compromise or bypass of authentication and access control mechanisms, destruction or compromise of data, or denial of service. Attack Scenario rules are in the definition stage for NADIR.

7 Results

The NADIR working prototype has been in operation since June of 1990. During this time NADIR identified and aided in the investigation of invalid activity by unknown users, and in the investigation of many cases of misuse or suspicious behavior by insiders. It has helped identify unanticipated network vulnerabilities, that have been remedied where possible or are being closely monitored. NADIR development has resulted in the identification of unanticipated misuse conditions, that have led to the definition of new expert rules. It has supported background analyses during investigations of several current and past ICN users. NADIR has also supplied unanticipated network management benefits. It has enabled us to detect hardware and software problems with some nodes of our network. It has also supplied detailed, statistical reports of network activity that were useful in such areas as accounting and network planning.

8 Future Directions

Anomaly and event notice now consists of terminal messages and periodic reports. For serious security events, the ultimate goal is to give notice on a near realtime basis.

Some kinds of invalid user activity, if allowed to continue, could lead to break-ins or denial of service to legitimate users. As a result, another goal is the notification of the proper ICN node of extremely suspicious activity, and the development of effective responses by that node. This would consist of taking direct action to stop an identified penetration attempt. The node's actions must be proportional to the extent that the monitored activity has deviated from valid behavior, what damage could result from allowing an invalid activity to continue, and denial of service considerations. We have not determined the criteria for such a response.

Finally, we would like to identify and use a rigorous method by which to validate and verify the performance, consistency, and completeness of the NADIR expert rule base.

9 Summary

NADIR shows the feasibility of the automation of security auditing on a distributed environment such as the ICN, and the benefits of applying an expert system to the problem. It shows the benefits of a phased approach to applying intrusion detection in a distributed environment. The working prototype is a start to a longer-range goal of expanding the system to more ICN nodes, and correlating their information to produce complete profiles of user activity on the ICN.

10 Acknowledgments

We wish to acknowledge the contributions of Jimmy McClary, who introduced us to the basic ideas, organized our funding, contributed enormously to our expert rule base, and supported us throughout the project. Valuable contributions to our rule base were made by members of the Operational Security Division. We are indebted to Harry Martz for his knowledge of statistics, and to Steve Ruud and Dorothy Merrigan for their contributions to the implementation of the NADIR system.

11 References

[1] D. Denning and P. Neumann. *Requirements and Model for IDES - A Real-Time Intrusion Detection Expert System, Final Report* (Computer Science Laboratory, SRI International, August 1985).

[2] M. Freiling, J. Alexander, S. Messick, S. Rehfuss, S. Shulman. *Starting a Knowledge Engineering Project: A Step-by-Step Approach* (The AI Magazine, Fall 1985).

[3] D. Denning. *An Intrusion Detection Model* (Proceedings of the IEEE Symposium on Security and Privacy, April 1986).

D. Denning. *An Intrusion Detection Model* (IEEE Transactions on Software Engineering, Vol. 13, No. 2, February 1987).

[4] D. Denning, D. Edwards, R. Jagannathan, T. Lunt, P. Neumann. *A Prototype IDES: A Real-Time Intrusion Detection Expert System* (Computer Science Laboratory, SRI International, August 1987).

[5] T. Lunt and R. Jagannathan. *A Prototype Real-Time Intrusion-Detection Expert System* (Proceedings of the IEEE Symposium on Security and Privacy, April 1988).

[6] T. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D. Edwards, P. Neumann, H. Javitz, A. Valdes. *IDES: The Enhanced Prototype A Real-Time Intrusion Detection Expert System* (SRI International, October 1988).

[7] M. Sebring, E. Shellhouse, M. Hanna, R. Whitehurst. *Expert Systems in Intrusion Detection: A Case Study* (Proceedings of the 11th National Computer Security Conference, October 1988).

[8] L. Halme and B. Kahn. *Building a Security Monitor with Adaptive User Work Profiles* (Proceedings of the 11th National Computer Security Conference, October 1988).

[9] T. Lunt. *Real-Time Intrusion Detection* (Proceedings of COMPCON, Spring1989).

[10] T. Lunt, R. Jagannathan, R. Lee, A. Whitehurst. *Knowledge-Based Intrusion Detection* (Proceedings of the 1989 AI Systems in Government Conference, March 1989).

[11] G. Tsudik and R. Summers. *AudES - An Expert System for Security Auditing* (Proceedings of AAAI Conference on Innovative Applications in AI, May 1990).

[12] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. *A Network Security Monitor* (Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1990).

[13] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neuman, C. Jalali. *IDES: A Progress Report* (Proceedings of the 6th Annual Computer Security Applications Conference, December 1990).

[14] J. Winkler. *A UNIX Prototype for Intrusion and Anomaly Detection in Secure Networks* (Proceeding of the 13th National Computer Security Conference, October 1990).

[15] K. Jackson, D. DuBois, and C. Stallings. *A Phased Approach to Network Intrusion Detection* (Proceedings of the DOE Computer Security Group Conference, May 1991, LA-UR-91-334).

FORMAL VERIFICATION OF A NETWORK SECURITY DEVICE: A CASE STUDY

Hicham N. Adra
CGI Information Systems
& Management Consultants
275 Slater Street
19th Floor
Ottawa, Ontario, Canada
K1P 5H9
Tel: (613) 234-2155
FAX: (613) 234-6934

William Sandberg-Maitland
CGI Information Systems
& Management Consultants
275 Slater Street
19th Floor
Ottawa, Ontario, Canada
K1P 5H9
Tel: (613) 234-2155
FAX: (613) 234-6934

ABSTRACT

An automated formal verification study of a commercial network security device, the SmartCrypto™, is described. A high level view of relevant formal verification techniques using the m-EVES environment is given. A description of the SmartCrypto™ is provided, as well as a brief overview of the m-EVES system. The uses and roles of Verification plans, environmental and device-specific models, and other planning techniques are discussed in the context of this case. Observations are made concerning the proof process and the problem of tractability which may apply to similar projects.

1.0 INTRODUCTION

This paper completes and extends the work initially reported in [ADRA91]. It describes some aspects of the formal verification of a commercially available Network Security Device (NSD). The NSD under study was the SmartCrypto™ of the CryptoNet™ product line by Intellinet. The study involved a selection of several source code modules, and the development of a formal verification of these target modules against specified properties using the m-EVES environment, described below. The purpose of this study was to establish that basic properties of functionality and security hold for the NSD design. This account extends that of [ADRA91], promoting the view of formal verification as a software engineering process. The conclusions attempt to summarize and generalize the experience gained. As in [ADRA91], technical details are suppressed.

The paper begins with a presentation of the target NSD system and a brief overview of the verification environment. A brief treatment on the theory of operations of the NSD is given. A more comprehensive treatment can be found in [ADRA91]. A description of the m-EVES verification environment follows. The emphasis is on the user view of the environment, rather than its internal design or technical details. The use of a verification plan is covered, with examples from the project. The role of modelling techniques is an important aspect of formal verification. Examples of modelling drawn from the project are discussed. The paper includes sections on the proof process, some techniques found to be of interest in this domain, and a set of general observations on formal verification issues. The summary draws together some opinions of the authors based on their experience. The role of formal verification within the context of the systems design and development process is highlighted.

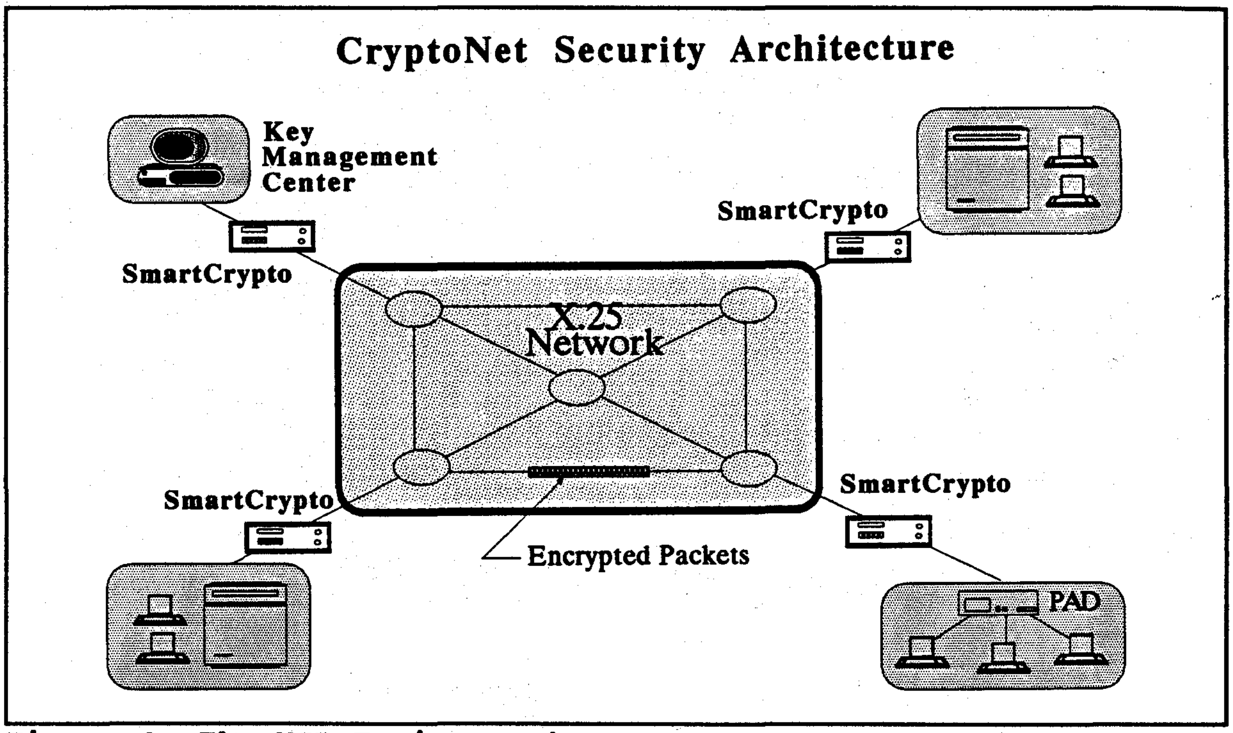


Figure 1: The NSD Environment

2.0 NSD DESCRIPTION

2.1 Theory of Operations

The NSD operates as an end-to-end encryption device functioning in an X.25 packet switching network [X.25]. The DES (Data Encryption Standard) is used in Cipher FeedBack (CFB) mode to achieve confidentiality of the information in the User Data field of X.25 data packets. The NSD is located between the Data Terminal Equipment (DTE) and the Data Circuit-terminating Equipment (DCE) or network. The NSD filters the traffic between the "host" and the network (see Figure 1). The host may be a computer system or a collection of terminals that are connected to an X.25 PAD (Packet Assembler/Disassembler). The Key Management Center (KMC) is responsible for the management of the network, including the distribution of keys and the remote monitoring and control of the network sites.

Several internal states are supported, including a Secure Normal State where encryption/decryption processing is performed on all data packet traffic between protected hosts. Since the control or header information is transmitted in its plaintext form and remains accessible to the intermediate nodes in the network, this method of applying encryption is called end-to-end; the nodes within the network do not need to be trusted to protect the security or secrecy of the information, as discussed in Section 4.

2.3 Security Protocol and Communication Requirements of NSD

Terminology related to CCITT Recommendation X.25 [X.25] will not be defined in detail here. Some discussion of X.25 issues is necessary to form the context for the inter-relation between security and communications functionality.

The NSD operates at the Network layer of the OSI model. Within the X.25 packet level DTE/DCE (interface (Level 3) frame of reference, the NSD acts as a data filter which intercepts data packets in either the DCE to DTE, or the DTE to DCE direction, and transforms them by decryption or encryption of user data fields. Most X.25 control packets are also recognised.

A Security Protocol is followed when a DTE call request packet initiates the setting up of a call. The NSD ensures that certain requirements are met; the security protocol relates to such elements of the X.25 call as the addresses, the logical channel numbers, the NSD network security groups, the encryption variables, and a security checksum.

Similar functions are performed in the case where the call is incoming from the DCE.

3.0 VERIFICATION TOOLS: m-EVES SYSTEM

The principal software tool used in this study was the m-EVES version 4 formal verification system running on a Sun 3/80 workstation under Sun OS version 4.0.3. m-EVES (an Environment for Verifying and Evaluating Software), of Odyssey Research Associates, is a prototype formal verification system [CRAI88] [EVES89]. m-EVES contains two main components:

- m-Verdi, a specification and implementation language,
- m-NEVER, an interactive/automated theorem prover.

An automated style of proof is possible with certain high level commands which invoke proof heuristics. m-EVES maintains an internal database of proven theorems. m-EVES has a soundness proof for its logic [EVES89]. For a more complete description of the m-EVES environment, see [CRAI88].

3.1 m-EVES Proof and Verification

This section defines some verification terminology referred to in the rest of the paper. A more detailed view of these topics will be found in [EVES89].

The term 'formula' will refer to a first order logical expression which is obtained from entering an m-Verdi target text in an m-EVES session. Informally, m-EVES reliably translates m-Verdi text into an equivalent and purely logical format which contains no occurrences of commands or other algorithmic language constructs. The resulting (initial) formula may be transformed into other formulas, using EVES Command Language (ECL) commands. Two particular formulas are of note: TRUE, which designates the universally valid formula, and FALSE, its negation.

Informally, a proof in m-EVES is a successful attempt to reduce the initial formula for a given m-Verdi target text to TRUE, through a finite number of steps. This generates a sequence of formulas, each formula derivable from its predecessor by application of an ECL command, and satisfying the following conditions:

- the first formula is obtained by m-EVES from the m-Verdi target,
- the final formula is TRUE.

The terms 'formula' and 'proof' are used in this paper exclusively in the sense given above.

3.2 Application of m-EVES to the NSD Verification Study

The NSD study involved a code verification of sample modules of the NSD system. As both specifications and implementation were known, a method was required to translate both into m-Verdi in the most reliable way. The implementation source code for the NSD is in the C language (with some assembler code). A number of hardware components, such as the encryption chip, also needed representation. Fortunately, the translation of target C code to m-Verdi was a relatively efficient manual operation. Some other software may not be as easily translatable, however, due to the use of pointers in C code which have no built-in support in m-Verdi. Hardware and environmental elements were translated by modelling techniques discussed below, and by the use of the m-Verdi "environment" construct.

A standard theory of history sequences is easily implemented in m-Verdi from examples in the literature [CRAI88]. The NSD study required some form of discrete temporal reasoning to prove that basic liveness properties hold. A simple history sequence theory is needed for this. Any system whose specifications include time dependencies between events will likely need a similar model. It was observed that certain theories expressed in terms of temporal logic have a natural embedding into first-order m-Verdi theories involving history sequences (see [FVR]).

This could be exploited in many general contexts.

Practicality may necessitate controlled modifications to the m-Verdi code. An example from this study illustrates this point. One of the target modules had interfaces with a large number of lower level sub-modules, each of which had extensive logical structure involving low level variables lying outside the general context of the verification module. In order to minimize the impact that such a code structure can have on the proof of high level structures, the low level modules were stubbed. This involves declaring them in m-Verdi without code, but possibly with logical annotations which describe their action. In addition, an array of flags, called an *occurrence array*, was defined. If a module is invoked, the boolean flag pertaining to it is set in the occurrence array. This can be done through specifying postconditions on the stubs.

Using this technique, it is possible to express general code-oriented specifications that say that under specific conditions, certain modules should be invoked. Proving this kind of specification provides assurance that the right sub-modules were called under various sets of conditions, and avoids the difficulty of contriving equivalent expressions employing low level variables. It is possible to return to the proof later and integrate the low level modules into the existing proof in a top-down manner, or employ some independent method to verify them. The use of in-line annotations, supported in m-Verdi as the "note" command, is also applicable to this problem.

4.0 VERIFICATION PLAN

4.1 Purpose of the Verification Plan

The role of the verification plan was to describe the scope of the verification effort in terms of an identification of general properties or areas of functionality within the NSD that were considered to be suitable objects of investigation in the next phase(s) of the project. The verification plan followed a phase where the architecture of the NSD, including the theory of operation, the hardware and software structure, and the functional requirements were analyzed. The next phase of the project was expected to involve the development of a design-level description of the verification targets, or modules to be specified, verified, and implemented in the context of the m-EVES environment.

The main focus in the verification plan was on the selection of a subset of the NSD for the purposes of formal verification and on the identification of some of the main issues that characterized the technical concerns at that stage. The criteria that guided the selection of verification targets included: the need to define the scope of effort based on the available resources, the importance of choosing important/critical properties of the device, and the characteristics of the formal verification environment and process.

4.2 Selection of Verification Properties

The selection of properties for formal verification is of critical importance to the value and level of achievement of the project as a whole. If the properties chosen reflect an overly simplistic or trivial view of the system, little is achieved in formally verifying them. If, on the other hand, the properties are either complex or inconsistent, the prospects for obtaining positive results become minimal or nonexistent. In the interest of obtaining useful results from the verification process, a practical balance between meaningful system properties and provable verification goals was the prime motivation for this task.

The criteria that were used for selection of the functionality that would be addressed can be broadly expressed as the following two areas:

- Security properties and
- Basic Functionality properties.

It is worth recalling that the NSD implements end-to-end encryption in an X.25 network [X.25]. It acts as a "filter" and is situated between the DTE or host and the DCE or network.

The properties selected and some necessary assumptions regarding the system are described in the following sections.

4.2.1 High Level Network Security Properties

The high level security properties of the NSD are informally based on the main aspects of information security. In this section these properties are discussed briefly and related to the major components of information security.

Security is generally considered to encompass the following three areas: Confidentiality, Integrity, and Availability. Threats to confidentiality relate to the unauthorized disclosure of information. Integrity refers to the properties through which the system or information meets one's expectations. Availability can be viewed in terms of the manifestation of its absence in the form of denial of service.

For the NSD, confidentiality is achieved through encryption. If encryption of data can be verified where it is required, then the confidentiality of data in the network is guaranteed. The property is largely dependent on the NSD processor which sends the data to the network. The assumption is made that no decryption activity can occur other than within another NSD. It is assumed that a DES-encrypted data packet cannot be read unless the key and initialization vector are known. With these assumptions, confidentiality is largely a byproduct of the basic CSP (see [HOAR85]) specifications that were developed during the Architecture Review stage. The specifications enforce rules regarding the conditions under which a data packet is encrypted. Confidentiality is compromised only if a data packet is not encrypted according to these rules.

While encryption contributes to a limited form of integrity, the data exchanged over the network can be manipulated and additional measures (at a higher communications layer) may be required to protect against threats to integrity. The use of distinct initialization vectors for each direction of an X.25 call aids in the detection of reverse direction replays. Also, the NSD supports a method for the authentication of the identities of the communicating parties. Through the logical design of the network and the ownership of the keys, an implicit form of authentication is achieved.

4.2.2 Basic Functionality Properties

The basic functionality of the NSD concerns properties relating to its role as a type of data encryption filter in a X.25 network, as well as the internal features which support this role, in particular, its security protocol. The term "basic functionality" is intended to describe the major NSD documented specifications around which the system is designed. In some cases, problems resulting from conflicts between hardware, communications and security requirements resulted in non-trivial modifications to the network layer behaviour of the NSD. It was important to determine that the NSD implementation actually satisfies these requirements.

The basic functionality of the NSD was primarily expressed in the formal specifications. These were written in CSP and required reliable translation to m-Verdi. This immediately implies that the underlying models (and their m-Verdi theories) must be compatible with whatever form the translations of the CSP functionality requirements take. A theory in m-Verdi which adequately describes the input-output black box view of the NSD must therefore mimic the behaviour specified in the CSP formal specifications.

4.2.2.1 X.25 Protocol Properties

The basic X.25 protocol is embedded in the CSP formal specification of the NSD. It was observed during the architecture analysis that not all internal X.25 states are implemented in the NSD, and that some events are not treated in the expected way. The specification took much of this into account, although the Call Collision State is present in this specification, but is not implemented in the NSD (since the encryption process does not allow for this). It was recognized that some modification of the specifications may therefore be in order prior to the verification activities.

Proving that the NSD satisfied a modified subset of X.25 was one of the primary objectives of the verification tasks.

4.2.2.2 NSD Security Protocol Properties

The security protocol for the NSD is embedded in the lower invocation levels of the CSP formal specification, i.e. in the form of special processes which are triggered when certain security-sensitive events occur. The main areas include Call Initialization, encryption-related events, and handling the interaction with the KMC. The last area was not considered to be within the scope of the verification plan.

Important high level properties are based on the sufficiency of the NSD security protocol. However, the strategy envisioned in the verification plan was not to establish the security protocol properties and then prove the high level properties. Rather, the NSD security protocol was seen as an inseparable part of the formal specification of the NSD. Its verification was seen as part and parcel of the verification of the basic functionality of the system. It would be possible in any case to draw on specific security properties obtained in the basic functionality verification in the establishment of high level security properties.

5.0 ROLE OF DEVICE-SPECIFIC MODELLING TECHNIQUES

Given a device with the level of complexity of the NSD, special models are often required to form a framework for stating certain specifications. Typically these models portray or simulate an environmental factor which the system must tolerate, a special theory or recognized standard which the system must satisfy, or possibly a subsystem or external entity whose characteristics are assumed and whose verification is considered beyond the scope of the project.

Implementation of a model in the verification language (e.g., m-Verdi) involves a design phase where decisions are made regarding the type and number of variables and data structures required. Some procedures may be designed and coded. In addition, a model will normally require purely logical components such as axioms and specification functions. A prototyping phase may be called for, in order to resolve design decisions. To finalize the initial model-building phase, theorems involving model constructs and relating them to the appropriate system interfaces are developed. Again, this may entail prototyping, as new model features may arise out of the attempt to prove the target theorems. In this way the body of theory involving the model is built up to the required level of depth.

Experience gained in this research indicates that even very simple models can entail significant costs in terms of time and effort over the verification phase. The effect of incrementally adding new models to a stable (i.e. proven) body of modules introduces the obligation to integrate all new variables and data structures into the old module proofs, and thus multiply their length. As more models are integrated in this way, the effect appears to be significantly non-linear. Although too little quantitative evidence is available at this point, this growth effect may have a significant influence on the design and scope of verification projects similar to the one documented here.

The following three main models were required by the verification phase:

- **Nondeterminism model**
  A model which allows proof of certain fault tolerance and liveness properties of the NSD under uncertainty of success of certain packet processing tasks.

- **X.25 model**
  A decision tree model of the state transitions of the X.25 protocol.

- **Encryption model**
  An elementary DES encryption (CFB) model based on axioms obtained from [FIPS81].

In each case, challenges were encountered with the integration of the new model into the existing proof database.

6.0 THE PROOF PROCESS

6.1 Verification of an Existing System

The nature of this project, which has its basis in an existing system, does not lend itself to the traditional top-down approach. In a general verification project one may have control over models of both the specification and the implementation and the verification can involve the development of parallel or corresponding descriptions. Attempting to gain assurance about a system after it has been developed through formal verification entails great difficulties. Verification of a low level of specification (eg. the source code) against a higher level of abstraction (eg. specifications of the requirements) is almost impossible without intermediate levels of detail. Models that characterize the specifications of system behaviour and the implementation must be developed. The specifications and the implementation must share various correspondences that relate to their logical structure and to their semantical content. The process of verification entails building the specification, the proof, and the implementation in tandem. For large systems this process is not easily managed, especially when an existing system is being examined. The implications for the certification of systems are serious.

6.2 Interaction with the Prover

Although provers such as that of the m-EVES environment are called automated provers, it should be remembered that they function as proof checkers and interactively assist in the proof process. The developer or user should ensure that the module or software being verified has been designed and implemented such that a proof would be forthcoming, and that the necessary conditions are met. The user will issue commands to effect certain proof steps, including the application of types of heuristics that the prover supports and the incorporation in the current proof of other theorems, lemmas, or assumptions. The user has to read the output of the prover at each step of the proof and be able to determine what parts of the current formula are of interest. The ability to see where conditions need to be strengthened or inconsistencies addressed is central to the process.

6.3 Modules and Preconditions

The verification of a module will show that if it is invoked when its precondition is satisfied it will terminate and its postcondition will be true. Even when the requisite proof is completed there remains the obligation to show that the precondition is satisfied in the calling module. Depending on the structure of the system and the time relationships between variables it may become very difficult to reason about the dependencies and to ensure the consistency of the various conditions when changes are made. When modules do not have side-effects and their preconditions are very simple this difficulty is reduced.

6.4 Use of Small Steps and Automation

The m_EVES prover has a number of 'macro' commands that may apply several basic or simple steps. These macro commands effect highly automated manipulations of a formula in an attempt to show that the condition it embodies is true. While it is desirable and sometimes easier to invoke these powerful commands to arrive at a proof, in cases where the formula is a complex or long logical condition the highly automated capabilities of the prover were not found to be very effective. The time required for a powerful command to execute becomes too long and the prover cannot be guaranteed to find a proof. The interactive application of a larger number of smaller steps was found to be more productive and allowed the developer greater flexibility in finding a proof. Although this required closer examination of the formula being verified at each step of the proof and was a very demanding process the ability to gradually simplify the formula and the higher likelihood of arriving at a proof made such a strategy necessary. Such a strategy is especially needed when an existing system is being studied since the verification team has less control over the software structure and design and the proof has to be adapted to the general architecture of the system.

6.5 Iterative Flow of Activities

The general view of formal verification as a top-down process of defining specifications and then implementing the software that demonstrably satisfies the specifications as evidenced by the proof is an abstraction in search of a reality. Formal verification necessitates a close match between the specifications and the implementation and between various parts of each. Changes to any part of the system descriptions may require changes to other parts depending on the dependencies that exist. Since it is unlikely that initial descriptions will be complete, changes and extensions must result. In an automated environment a considerable amount of the formal descriptions or theories are developed to support the verification effort or in order for the prover to deal with leaps of abstraction and are not strictly part of the system. Thus the likelihood of the discovery of the need for additional assertions, properties, and relationships is very high, and semantical changes often require substantial concern with aspects of the verification environment and language. The result is a process that defies simplistic depictions and which requires both anticipation of what is needed and the ability to recognize that changes will be necessary. A developer should expect a considerable amount of both planning and refinement.

6.6 Gradual Building of Proof Requirements

Despite the high penalty for changes to a system's description (specification or implementation), in some cases it is useful to experiment with a module to arrive at the proper form and conditions. In such cases the weaker or simpler forms of the (post) condition that must be shown may be used to gain confidence in the correctness of the module's code or structure and to quickly discover any flaws, which would be easier to detect since any inconsistencies will be more apparent in the simpler formula. The general strategy that was used in the proof may also be applicable to the stronger or final condition.

7.0 OBSERVATIONS AND FINDINGS

7.1 Meaning of Verification Results

The mathematical nature of formal methods and the benefits of (automated or computer assisted) formal verification do not obviate the need for a critical assessment of the level of assurance provided by the verification effort. The system descriptions and documentation -- in the form of such constructs as axioms, data declarations, executable code, and theorems -- may encompass several assumptions and models, the appropriateness or validity of which cannot be determined solely within the steps and proofs of the formal verification effort or environment.

The formal verification results may not guarantee the absence of inconsistencies in the system specifications. In addition, the strength and completeness of the assertions that are shown is central to the value of the verification effort. The verification team is free to choose the form of a module and the statements of the precondition and postcondition. The assurance, about the behaviour of a module, that is provided by a proof (based on the precondition and postcondition) is not always clear, especially with respect to intermediate occurrences of conditions and states. The role of the results of an individual proof must be carefully considered in relation to other proofs and within the general description of the system. The results of the verification effort as a whole, in turn, must be assessed with a recognition of any assumptions and limitations that exist and to determine the implications for the behaviour and trustworthiness of the system.

7.2 Engineering Side of Formal Verification

In addition to the mathematical nature of formal verification, the development of formal and executable specifications and descriptions involves many of the choices and decisions that characterize engineering processes. Many of the engineering issues do not manifest themselves in the formal verification of a simple or small application, partly because the implications of the decisions may not be critical to the success of the software effort. When large and complex systems are being built, however, the number and difficulty of the choices that the project team faces are increased. The quantitative aspects may become qualitative in that gradual accumulation of complexity may represent unsurmountable obstacles to the successful completion of the project. The ability to operate within an integrated project support environment is expected to form a key requirement of formal specification and verification tools.

While most likely there is no general recipe for building systems using formal methods and automated verification environments, there are various approaches that are effective in dealing with the complexity that faces software designers and developers. Although it is beyond the scope of this paper to describe design or development methods, some of the observations in this section may hint at some properties that such methods should have.

7.3 Relative Size of the Formal Descriptions

In specifying and implementing modules in m-Verdi it was observed that the size of the resulting software was considerably larger than the original C source code. This may reflect a basic difference between software development that uses third generation languages and that which is based on formal verification. In the latter case, some of the information that would traditionally reside in the various requirements and design documents has to be represented in the formal specifications that are developed within the verification environment. Also, formal verification may require or be facilitated by the specification of supporting models and theories that capture the functionality of the software and bridge the gaps between different levels of abstraction. The general observation was that the size of the formal product far exceeded the C language source code. The use of graphical depiction techniques may aid in addressing the difficulties associated with the long expressions and formulas; graphical nested structures may be presented to the analyst in order to increase the communications bandwidth of the user interface between the environment and the user [TAR]. However, the implications for a large project of the large size of formal descriptions are very serious when the effect of software size on the required effort and schedule, as discussed below, is recognized.

7.4 Effect of System Growth on Schedule and Effort

The relationship between software size and the effort or time required to develop that software is considered to be non-linear, and in fact many estimation models represent effort as an exponential function of size. In this project, the addition of more functionality and the attempt to integrate these modules with the existing system descriptions required considerable effort. The nature of formal verification as an effort and time intensive process and the need for ensuring proof consistency indicate an even steeper form of the curve for effort as a function of system size. The consequences for the direct application of formal verification to large projects are serious and seem to entail considerable limitations.

7.5 Intermediate Levels of Abstraction

While the step-wise refinement of system descriptions is a generally known technique for design and development, the use of specifications at different levels of abstraction has special implications for formal methods and for attempts to achieve a high level of assurance as required in secure systems. The use of a several levels of abstraction facilitates the mapping between levels. It also increases the effort required to manage the system descriptions and may become less effective when too many levels are used. In formal verification efforts the project team may find the development of intermediate specifications (or implementation layers) to be necessary in interacting with an automated prover which can not be expected to deal with wide abstraction gaps. The recognition of: the need to maintain the consistency of the software descriptions, performance considerations, and the ripple effect of changes to one module on other parts suggests a trade-off between these factors and the number of levels in the abstraction hierarchy. The careful introduction of intermediate layers remains an effective strategy for simplifying formal proofs.

7.6 Propagation of Properties to Higher Levels of Abstraction

As modules at the lower levels of the module hierarchy, including those that have a relatively self-contained function, are developed and integrated with modules at a higher level their properties need to be reflected in the properties of the upper layers. This process of making the function of lower-level modules known to a higher module involves the upward migration of properties within the hierarchical structure of the software. The preconditions and postconditions at adjoining layers need to be closely linked. Even after achieving the proper correspondence between the modules and their properties, there is a strong likelihood that changes to a lower level will be necessary and that a large part of the time that was expended in the verification of the existing modules will be required again.

7.7 Understanding the Automated Prover Output

In interacting with an automated prover, the software developer or user is faced with several characteristics of that tool and environment. One such aspect is the length and complexity of verification conditions that are generated by the prover. The automated capability of the prover includes the generation of a formula that formally describes the module and its corresponding proof obligation and the execution of user commands that represent steps in verifying that the formula is true. The formulas may be long and complex. The developer has to read the formula and determine what steps are necessary for its proof or whether a proof can be found at all; the formula may be amenable to proof with the right sequence of commands, or it may be inconsistent or lacking some necessary assertions in which case it cannot be proven.

Thus, although the prover automates certain capabilities the user person has to interact with it and is expected to 'process' its output. This seems to tie the person to the technological tool. It is not clear to what extent expecting a developer to read the long and intricate product of a tool (at every intermediate step) represents the best form of an activity's automation. This may be one stage in the evolution of verification technology and further advances in software engineering may be expected to alter this cooperative process.

7.8 Focus on Critical Functionality

For several reasons that include the observed growth effect in terms of the relationship between the size of the system descriptions and the required effort, as described earlier, the choice of the appropriate scope for formal verification is considered to be essential to the success of a formal verification project. It is important to limit the targets of formal verification to the "critical" functionality or components of the system under consideration. While determining the critical nature of a component is a matter of judgement and may be based on the area of interest or depend on the design of the system, the choice of the proper scope for formal verification is necessary to the management of the complexity of the verification process.

8.0 SUMMARY

This paper has identified several aspects of formal verification as applied to a network security device. The use of certain modelling techniques was described in the context of the goals of the verification effort. The role of a verification plan within the life cycle of a formal verification project was shown through a case study approach to the communication of findings. The results and observations described in this paper, and in Sections 6 and 7 in particular, were part of this attempt to share the project team's general experience in the application of formal verification to a target having substantial scope of functionality.

In the verification of the NSD, the general areas that were addressed included, in addition to the general X.25 functionality, both safety and liveness properties. Safety properties within the context of the NSD were examined in terms of the controlled application of encryption. The use of a model of nondeterminism to support liveness stemmed from a choice to characterize the rich behaviour of a communications system and to avoid extreme simplifications. While it is recognized that abstraction is an essential part of the use of formal methods and is often necessary in describing systems, the need remains for formal verification efforts to address the complex or flexible behaviour that is inherent in some systems.

The logical separation between "security" and "functionality" or "liveness" is sometimes detrimental to the evaluation of a system. (Among other limitations: it seems to reflect a bias towards viewing security as secrecy or confidentiality. Furthermore, the assumption that properties of a system are independent or even that different areas of security can be examined in isolation is not justified.) While a system that does nothing can be considered safe with regards to confidentiality its trustworthiness is of very little value. It is in the area of complex systems and rich behaviour that security and trust are of special interest and importance.

The relationship between the general functionality of a system and its security policy, especially in their implementations, often involves many subtle dependencies and provides a challenge in any attempt to gain the high degree of assurance that is necessary for the system to be considered trustworthy. The formal proof process and the use of an automated environment contribute to both the challenges and the solutions. The tractability of the verification problem, as manifested by the effort and time required, is a serious concern. It is hoped that this paper will contribute to the recognition of the role of formal verification and specification within the systems engineering process and of the need for verification design methods and techniques that support the management of the complexity of the formal development process. Towards such a view, it is important to recognize the role, limitations, and benefits of formal verification within an integrated systems design and development process.

ACKNOWLEDGEMENTS

The work described in this paper was funded by the Canadian Department of National Defence under contract No. W8477-9-CC07/01-QD. The support and insights of Mr. Milan Kuchta and Mr. Vincent Taylor are gratefully acknowledged. This paper has also benefited from the comments and suggestions of the referees. The authors would also like to thank Intellinet Corp. for their help and for providing access to the CryptoNet^{TM}/SmartCrypto^{TM} technical documentation.

REFERENCES

[ADRA91] Adra, H.N. and Sandberg-Maitland, W., "Formal Verification Techniques for a Network Security Device", Proceedings of the Third Annual Canadian Computer Security Symposium, May 15-17, 1991.

[CRAI88] Craigen, Dan, "An Application of the m-EVES Verification System", Proceedings of the Second Workshop on Software Testing, Verification, and Analysis", IEEE, July 1988, pp. 21-36.

[EVES89] *m-EVES Collected Papers*, Odyssey Research Associates, Inc., Ottawa, Ontario, Canada, September 14, 1990.

[FIPS81] U.S. Department of Commerce/National Bureau of Standards, FIPS Publication 74, *Guidelines for Implementing and Using the NBS Data Encryption Standard*, April, 1981.

[FVR] *Verification of a Network Security Device: Final Verification Report (FVR)*, Technical Report SE-R91.009, CGI Group, Ottawa, Ontario, Canada, March 1991.

[HOAR85] Hoare, C.A.R., *Communicating Sequential Processes*, Prentice-Hall International, 1985.

[INT87] *CRYPTONET / SMART CRYPTO Theory of Operation and Functional Specification*, Technical Report INT-87-37, Intellitech Canada Limited, Ottawa, November 12, 1987.

[TAR] *Verification of a Network Security Device: Technical Assessment Report (TAR)*, Technical Report SE-R91.008, CGI Group, Ottawa, Ontario, Canada, March 1991.

[VPR] *Verification of a Network Security Device: Verification Plan and Rationale (VPR)*, Technical Report SE-R90.29, CGI Group, Ottawa, Ontario, Canada, May 1990.

[VR1] *Verification of a Network Security Device: Verification Report 1 (VR1)*, Technical Report SE-R90.43, CGI Group, Ottawa, Ontario, Canada, September 1990.

[X.25] *Interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit*, CCITT Recommendation X.25, 1984.

A FRAMEWORK FOR ADVANCING INTEGRITY STANDARDIZATION

Terry Mayfield
Stephen R. Welke
John M. Boone
Catherine W. McDonald

Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311
(703) 845-3500

Abstract

This paper deals with the issue of preserving and promoting integrity within computer and automated information systems. It is intended to serve as the starting point for defining those expectations and standardizing integrity properties of systems. The paper discusses the difficulty of developing a single definition of the term integrity as it applies to data and systems. Integrity has multiple definitions in the dictionary and the application of those definitions to data and systems using a single attribute within the expectation set has led to definitions that could not achieve consensus. Concluding that a single definition is not needed to advance our understanding, the paper develops a more appropriate operational definition, or framework, that encompasses various views of the issue. This framework includes the two distinct, yet interdependent, contexts for integrity: data and systems. The framework reinterprets, within these two contexts, a general integrity protection goal to derive three specific integrity goals. The framework also interprets the integrity properties and relationships of active and passive entities in a system using the conceptual constraints of "adherence to a code of behavior," "wholeness," and "risk reduction." The paper concludes that it is possible to begin to standardize integrity properties. We acknowledge that gaps in understanding exist, but recommend that further studies be undertaken. We conclude that such studies can be accomplished concurrently with standardization and that both efforts could be mutually supportive.

1. Introduction

As public, private, and defense sectors of our society have become increasingly dependent on widely used interconnected computers for carrying out critical as well as more mundane tasks, integrity of these systems and their data has become a significant concern. The purpose of this paper is not to motivate people to recognize the need for integrity, but rather to motivate the use of what we know about integrity and to stimulate more interest in research to standardize integrity properties of systems. This paper provides a framework for examining the issue of promoting and preserving integrity in computer systems. It is intended to be used as a general foundation for further investigations into integrity and a focus for debate on those aspects of integrity related to computer and automated information systems (AISs).

One of the specific further investigations is the development and evolution of product evaluation criteria to assist the U.S. Government in the acquisition of systems that incorporate integrity preserving mechanisms. These criteria also will help guide computer system vendors in

The work reported in this paper was conducted as part of Institute for Defense Analyses Project T-AA5-459 under Contract No. MDA903-89-C-0003 for the Department of Defense. It is based on portions of IDA Paper P-2316, *Integrity in Computer and Automated Information 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.

producing systems that can be evaluated in terms of protection features and assurance measures needed to ascertain a degree of trust in the product's ability to promote and preserve system and data integrity. In support of this criteria investigation, we have provided a separate document [1] that offers potential modifications to the Control Objectives contained in the Trusted Computer Systems Evaluation Criteria (TCSEC), DoD 5200.28-STD [2]. The modifications extend the statements of the control objectives to encompass data and systems integrity; specific criteria remain as future work.

2. Background

For some time, both integrity and confidentiality have been regarded as inherent parts of information security (INFOSEC). Confidentiality, however, has been addressed in greater detail than integrity by evaluation criteria such as the TCSEC. The emphasis on confidentiality has resulted in a significant effort at standardizing confidentiality properties of systems, without an equivalent effort on integrity. However, this lack of standardization effort does not mean that there is a complete lack of mechanisms for or understanding of integrity in computing systems. A modicum of both exists. Indeed, many well-understood protection mechanisms initially designed to preserve integrity have been adopted as standards for preserving confidentiality. What has not been accomplished is the coherent articulation of requirements and implementation specifications so that integrity property standardization can evolve. There is a need now to put a significant effort on standardizing integrity properties of systems. This paper provides a starting point.

The original impetus for this paper derives from an examination of computer security requirements for military *tactical* and *embedded* computer systems, during which the need for integrity criteria for military systems became apparent. As the military has grown dependent on complex, highly interconnected computer systems, issues of integrity have become increasingly important. In many cases, the risks related to disclosure of information, particularly volatile information which is to be used as soon as it is issued, may be small. On the other hand, if this information is modified between the time it is originated and the time it is used (e.g., weapons actions based upon it are initiated), the modified information may cause desired actions to result in failure (e.g., missiles on the wrong target). When one considers the potential loss or damage to lives, equipment, or military operations that could result when the integrity of a military computer system is violated, it becomes more apparent why the integrity of military computer systems can be seen to be at least as important as confidentiality.

There are many systems in which integrity may be deemed more important than confidentiality (e.g., educational record systems, flight-reservation systems, medical records systems, financial systems, insurance systems, personnel systems). While it is important in many cases that the confidentiality of information in these types of systems be preserved, it is of crucial importance that this information not be tampered with or modified in unauthorized ways. It is especially important that unauthorized tampering not occur in embedded computer systems. These systems are components incorporated to perform one or more specific (usually control) functions within a larger system. They present a more unique aspect of the importance of integrity as they often may have little or no human interface to aid in providing for correct systems operation. Embedded computer systems are not restricted to military weapons systems. Commercial examples include anti-lock braking systems, aircraft avionics, automated milling machines, radiology imaging equipment, and robotic actuator control systems.

Integrity can be viewed not only in the context of relative importance but also in the historical context of developing protection mechanisms within computer systems. Many protection mechanisms were developed originally to preserve integrity. Only later were they recognized to be equally applicable to preserving confidentiality. One of the earliest concerns in the development of computers was that programs might be able to access memory (either primary memory or secondary memory such as disks) that was not allocated to them. As soon as systems began to allocate resources to more than one program at a time (e.g., multitasking, multiprogramming, and time-sharing), it became necessary to protect the resources allocated to the concurrent execution of routines from accidentally modifying one another. This increased system concurrency led to a form of interleaved sharing of the processor using two or more processor states (e.g., one for problem or user state and a second for control or system state), as well as interrupt, privilege, and protected address spaces implemented in hardware and software. These "mechanisms" became the early foundations for "trusted" systems, even though they generally began with the intent of protecting against errors in programs rather than protecting against malicious actions. The mechanisms were aids to help programmers debug their programs and to protect them from their own coding errors. Since these mechanisms were designed to protect against accidents, by themselves or without extensions they offer little protection against malicious attacks.

3. Defining Integrity

Integrity is a term that does not have an agreed definition or set of definitions for use within the INFOSEC community. Recent efforts to define and model integrity have raised the importance of addressing integrity issues and the incompleteness of the TCSEC with respect to integrity. They also have sparked renewed interest in examining what needs to be done to achieve integrity property standardization in computing systems. However, the INFOSEC community's experience to date in trying to define integrity provides ample evidence that it doesn't seem to be profitable to continue to try and force a single consensus definition. Thus, we elect not to debate the merits of one proposed definition over another. Rather, we accept that the definitions generally all point to a single concept termed "integrity."

Our position is reinforced when we refer to a dictionary; integrity has multiple definitions [3]. Integrity is an abstract noun. As with any abstract noun, integrity derives more concrete meaning from the term(s) to which it is attributed and from the relations of these terms to one another. In this case, we attribute integrity to two separate, although interdependent, terms (i.e., *data* and *systems*). Bonyun made a similar observation in discussing the difficulty of arriving at a consensus definition of integrity [4]. He also recognized the interdependence of the terms systems and data in defining integrity, and submitted the proposition that "in order to provide any measure of assurance that the integrity of data is preserved, the integrity of the system, as a whole, must be considered."

Keeping this proposition in mind, we develop a conceptual framework or operational definition which is largely derived from the mainstream writing on the topic and which we believe provides a clearer focus for this body of information. We start by defining two distinct contexts of integrity in computing systems: *data integrity*, which concerns the objects being processed, and *systems integrity*, which concerns the behavior of the computing system in its environment. We then relate these two contexts to a general integrity goal developed from writings on information protection. We reinterpret this general goal into several specific integrity goals. Finally, we establish three conceptual constraints that are important to the discussion of the preservation and promotion of integrity. These definitions, specific goals, and conceptual constraints provide our framework or operational definition of integrity. A diagram of this framework is given in Figure 1.

3.1. Data Integrity

Data integrity is what first comes to mind when most people speak of integrity in computer systems. To many, it implies attributes of data such as quality, correctness, authenticity, timeliness, accuracy, and precision. Data integrity is concerned with preserving the meaning of information, with preserving the completeness and consistency of its representations within the system, and with its correspondence to its representations external to the system. It involves the successful and correct operation of both computer hardware and software with respect to data and, where applicable, the correct operations of the users of the computing system (e.g., data entry). Data integrity is a primary concern in AISs that process more than one distinct type of data using the same equipment, or that share more than one distinct group of users. It is of concern in large scale, distributed, and networked processing systems because of the diversity and interaction of information with which such systems must

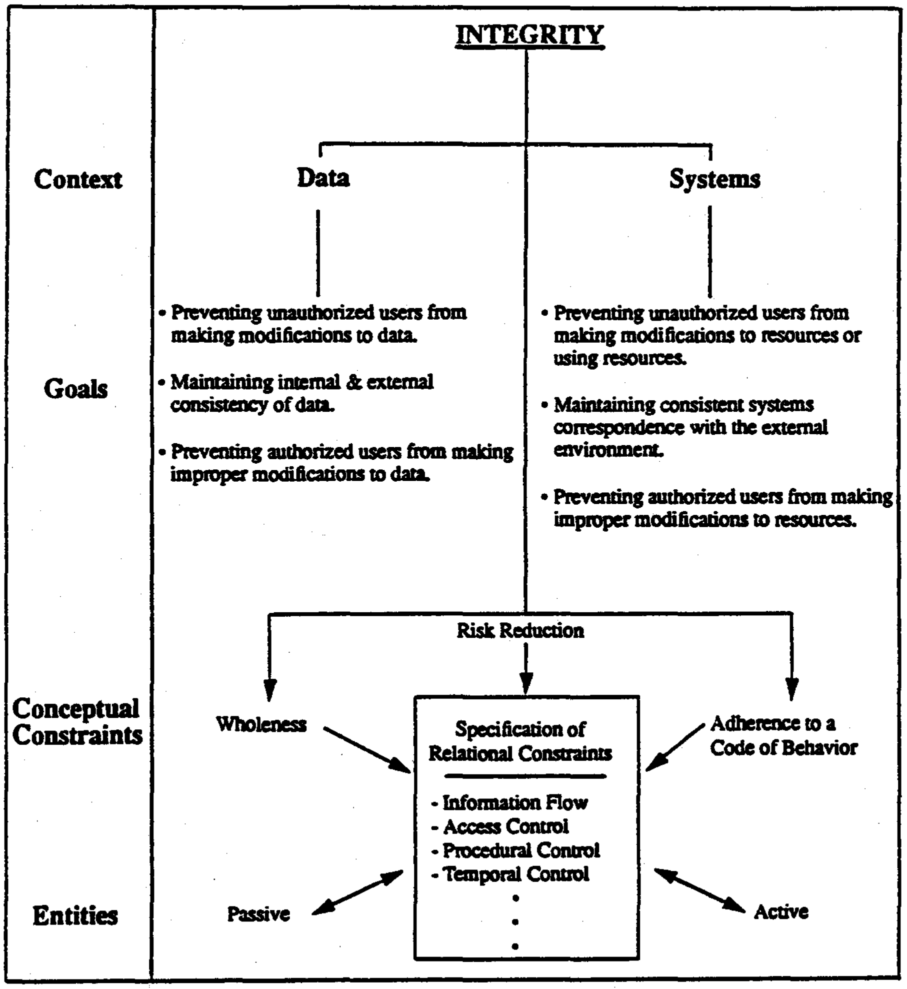

INTEGRITY
Context
Data
Systems
Goals
• Preventing unauthorized users from making modifications to data.
• Maintaining internal & external consistency of data.
• Preventing authorized users from making improper modifications to data.
• Preventing unauthorized users from making modifications to resources or using resources.
• Maintaining consistent systems correspondence with the external environment.
• Preventing authorized users from making improper modifications to resources.
Risk Reduction
Conceptual Constraints
Wholeness
Specification of Relational Constraints
- Information Flow
- Access Control
- Procedural Control
- Temporal Control
Adherence to a Code of Behavior
Entities
Passive
Active

Figure 1. Integrity Framework

often deal, and because of the potentially large and widespread number of users and system nodes that must interact via such systems.

3.2. Systems Integrity

Systems integrity is defined here as the successful and correct operation of computing resources. Systems integrity is an overarching concept for computing systems, yet one that has specific implications in embedded systems whose control is dependent on system sensors. Systems integrity is closely related to the domain of fault tolerance. This aspect of integrity often is not included in the traditional discussions of integrity because it involves an aspect of computing, fault tolerance, that is often mistakenly relegated to the hardware level. Systems integrity is only superficially a hardware issue, and is equally applicable to the AIS environment; an embedded system simply has less user-provided fault tolerance. In this context, it also is related closely to the issue of system safety (e.g., the safe operation of an aircraft employing embedded computers to maintain stable flight). In an embedded system, there is usually a much closer connection between the computing machinery and the physical, external environment than in a command and control system or a conventional AIS. The command and control system or conventional AIS often serves to process information for human users to interpret, while the embedded system most often acts in a relatively autonomous sense.

Systems integrity is also related to what is traditionally called the *denial of service* problem. Denial of service covers a broad category of circumstances in which basic system services are denied to the users. However, systems integrity is less concerned with denial of service than with alteration of the ability of the system to perform in a consistent and reliable manner, given an environment in which system design flaws can be exploited to modify the operation of the system by an attacker.

For example, because an embedded system is usually very closely linked to the environment, one of the fundamental, but less familiar, ways in which such an attack can be accomplished is by distorting the system's view of time. This type of attack is nearly identical to a denial of service attack that interferes with the scheduling of time-related resources provided by the computing system. However, while denial of service is intended to prevent a user from being able to employ a system function for its intended purpose, time-related attacks on an embedded system can be intended to alter, but not stop, the functioning of a system. System examples of such an attack include the disorientation of a satellite in space or the confusing of a satellite's measurement of the location of targets it is tracking by forcing some part of the system outside of its design parameters. Similarly, environmental hazards or the use of sensor countermeasures such as flares, smoke, or reflectors can cause embedded systems employing single sensors such as infrared, laser, or radar to operate in unintended ways.

When sensors are used in combination, algorithms often are used to fuse the sensor inputs and provide control decisions to the employing systems. The degree of dependency on a single sensor, the amount of redundancy provided by multiple sensors, the dominance of sensors within the algorithm, and the discontinuity of agreement between sensors are but a few of the key facets in the design of fusion algorithms in embedded systems. It is the potential design flaws in these systems that we are concerned with when viewing systems from the perspective of systems integrity.

3.3. Information System Protection Goals

Many researchers and practitioners interested in INFOSEC believe that the field is concerned with three overlapping protection goals: *confidentiality*, *integrity*, and *availability*. From a general review of reference material, we have broadly construed these individual goals as having the following meanings:

a. Confidentiality denotes the goal of ensuring that information is protected from improper disclosure.

b. Integrity denotes the goal of ensuring that data has at all times a proper physical representation, is a proper semantic representation of information, and that authorized users and information processing resources perform correct processing operations on it.

c. Availability denotes the goal of ensuring that information and information processing resources both remain readily accessible to their authorized users.

The above integrity goal (b) is complete only with respect to data integrity. It remains incomplete with respect to systems integrity. We extend it to include ensuring that the services and resources composing the processing system are impenetrable to unauthorized users. This extension provides for a more complete categorization of integrity goals, since there is no other category for the protection of information processing resources from unauthorized use, the *theft of service* problem. It is recognized that this extension represents an overlap of integrity with availability. Embedded systems require one further extension to denote the goal of consistent and correct performance of the system within its external environment.

3.4. Integrity Goals

Using the goal previously denoted for integrity and the extensions we propose, we reinterpret the general integrity goal into the following specific goals in what we believe to be the order of increasing difficulty to achieve. None of these goals can be achieved with absolute certainty; some will respond to mechanisms known to provide some degree of assurance and all may require additional risk reduction techniques.

3.4.1. Preventing Unauthorized Users From Making Modifications

This goal addresses both data and system resources. Unauthorized use includes the improper access to the system, its resources and data. Unauthorized modification includes changes to the system, its resources, and changes to the user or system data originally stored including addition or deletion of such data. With respect to user data, this goal is the opposite of the confidentiality requirement: confidentiality places restrictions on information flow out of the stored data, whereas in this goal, integrity places restrictions on information flow into the stored data.

3.4.2. Maintaining Internal and External Consistency

This goal addresses both data and systems. It addresses self-consistency of interdependent data and consistency of data with the real-world environment that the data represents. Replicated and distributed data in a distributed computing system add new complexity to maintaining internal consistency. Fulfilling a requirement for periodic comparison of the internal data with the real-world environment it represents would help to satisfy both the data and systems aspects of this integrity goal. The accuracy of correspondence may require a tolerance that accounts for data input lags or for real-world lags, but such a tolerance must not allow incremental attacks in smaller segments than the tolerated range. Embedded systems that must rely only on their sensors to gain knowledge of the external environment require additional specifications to enable them to internally interpret the externally sensed data in terms of the correctness of their systems behavior in the external world. It is the addition of overall systems semantics that allows the embedded system to understand the consistency of external data with respect to systems actions.

a. As an example of internal data consistency, a file containing a monthly summary of transactions must be consistent with the transaction records themselves.

b. As an example of external data consistency, inventory records in an accounting system must accurately reflect the inventory of merchandise on hand. This correspondence may require controls on the external items as well as controls on the data representing them (e.g., data entry controls). The accuracy of correspondence may require a tolerance that accounts for data input lags or for inventory in shipment, but not actually received.

c. As an example of systems integrity and its relationship to external consistency, an increasing temperature at a cooling system sensor may be the result of a fault or an attack on the sensor (result: overcooling of the space) or a failure of a cooling system component such as a freon leak (result: overheating of the space). In both cases, the automated thermostat (embedded system) could be perceived as having an integrity failure unless it could properly interpret the sensed information in the context of the thermostat's interaction with the rest of the system, and either provide an alert of the external attack or failure, or provide a controlling action to counter the attack or overcome the failure. The essential requirement is that in order to have the system maintain a consistency of performance with its external environment, it must be provided with an internal means to interpret and flexibility to adapt to the external environment.

3.4.3. Preventing Authorized Users From Making Improper Modifications

The final goal of integrity is the most abstract, and usually involves risk reduction methods or procedures rather than absolute checks on the part of the system. Preventing improper modifications may involve requirements that ethical principles not be violated; for example, an employee may be authorized to transfer funds to specific company accounts, but should not make fraudulent or arbitrary transfers. It is, in fact, impossible to provide absolute "integrity" in this sense, so various mechanisms are usually provided to minimize the risk of this type of integrity violation occurring.

3.5. Conceptual Constraints Important to Integrity

There are three conceptual constraints that are important to the discussion of integrity. The first conceptual constraint has to do with the active entities of a system. We use the term *agents* to denote users and their surrogates. Here, we relate one of the dictionary definitions [3] of integrity, *adherence to a code of behavior*, to actions of systems and their active agents. The second conceptual constraint has to do with the passive entities or objects of a system. Objects as used here are more general than the storage objects as used in the TCSEC. We relate the states of the system and its objects to a second of Webster's definitions of integrity, *wholeness*. We show that the constraint relationships between active agents and passive entities are interdependent. We contend that the essence of integrity is in the specification of constraints and execution adherence of the active and passive entities to the specification as the active agent transforms the passive entity. Without specifications, one cannot judge the integrity of an active or passive entity. The third system conceptual constraint deals with the treatment of integrity when there can be no absolute assurance of maintaining integrity. We relate integrity to a fundamental aspect of protection, a strategy of *risk reduction*.

3.5.1. Adherence to a Code of Behavior

Adherence to a code of behavior focuses on the constraints of the active agents under examination. It is important to recognize that agents exist at different layers of abstraction (e.g., the user, the processor, the memory management unit). Thus, the focus on the active agents is to ensure that their actions are sanctioned or constrained so that they cannot exceed established bounds. Any action outside of these bounds, if attempted, must be prevented or detected prior to having a corrupting effect. Further, humans, as active agents, are held accountable for their actions and held liable to sanctions should such actions have a corrupting effect. One set of applied constraints are derived from the expected states of the system or data objects involved in the actions. Thus, the expected behaviors of the system's active agents are conditionally constrained by the results expected in the system's or data object's states. These behavioral constraints may be statically or dynamically conditioned.

For example, consider a processor (an active agent) stepping through an application program (where procedural actions are conditioned or constrained) and arriving at the conditional instruction where the range (a conditional constraint) of a data item is checked. If the program is written with integrity in mind and the data item is "out of range," the forward progress of the processor through the applications program is halted and an error handling program is called to allow the processor to dispatch the error. Further progress in the application program is resumed when the error handling program returns control of the processor back to the application program.

A second set of applied constraints are derived from the temporal domain. These may be thought of as *event constraints*. Here, the active agent must perform an action or set of actions within a specified bound of time. The actions may be sequenced or concurrent, they may be performance constrained by rates (i.e., actions per unit of time), activity time (e.g., start & stop), elapsed time (e.g., start + 2hrs), and discrete time (e.g., complete by 1:05 p.m.)

Without a set of specified constraints, there is no "code of behavior" to which the active agent must adhere and, thus, the resultant states of data acted upon are unpredictable and potentially corrupt.

3.5.2. Wholeness

Wholeness has both the sense of unimpaired condition (i.e., soundness) and being complete and undivided (i.e., completeness) [3]. This aspect of integrity focuses on the incorruptibility of the objects under examination. It is important to recognize that objects exist at different layers of abstraction (e.g., bits, words, segments, packets, messages, programs). Thus, the focus of protection for an object is to ensure that it can only be accessed, operated on, or entered in specified ways and that it otherwise cannot be penetrated and its internals modified or destroyed. The constraints applied are those derived from the expected actions of the system's active agents. There are also constraints derived from the temporal domain. Thus, the expected states of the system or data objects are constrained by the expected actions of the system's active agents.

For example, consider the updating of a relational database with one logical update transaction concurrently competing with another logical update transaction for a portion of the set of data items in the database. The expected actions for each update are based on the constraining concepts of *atomicity* (i.e., that the actions of a logical transaction shall be complete and that they shall transform each involved individual data item from one unimpaired state to a new unimpaired state, or that they shall have the effect of not carrying out the update at all); *serializability* (i.e., the consecutive ordering of all actions in the logical transaction schedule); and *mutual exclusion* (i.e., exclusive access to a given data item for the purpose of completing the actions of the logical transaction). The use of mechanisms such as dependency ordering, locking, logging, and the two-phase commit protocol enable the actions of the two transactions to complete leaving the database in a complete and consistent state.

3.5.3. Risk Reduction

Integrity is constrained by the inability to ensure absolute protection. The potential results of actions of an adversarial attack, or the results of the integrity failure of a human or system component place the entire system at risk of corrupted behavior. This risk could include complete system failure, corrupted representations of data, or complete loss of data. Therefore, a strategy of protection which includes relatively assured capabilities provided by protection mechanisms plus measures to reduce the exposure of human, system component, and data to loss of integrity should be pursued. Such a risk reduction strategy could include the following:

a. Containment to construct "firewalls" to minimize exposures and opportunities to both authorized and unauthorized individuals (e.g., minimizing, separating, and rotating data, minimizing privileges of individuals, separating responsibilities, and rotating individuals).

b. Monitors to actively observe or oversee human and system actions, to control the progress of the actions, log the actions for later review, and/or alert other authorities of inappropriate action.

c. Sanctions to apply a higher risk (e.g., fines, loss of job, loss of professional license, prison sentence) to the individual as compared to the potential gain from attempting, conducting, or completing an unauthorized act.

d. Fault tolerance via redundancy (e.g., databases to preserve data or processors to preserve continued operation in an acknowledged environment of faults). Contingency or backup operational sites are another form of redundancy. Note: layered protection, or protection in depth, is a form of redundancy to reduce dependency on the impenetrability of a single protection perimeter.

e. Insurance to replace the objects or their value should they be lost or damaged (e.g., fire insurance, theft insurance, and liability insurance).

4. Conclusions & Recommendations

This paper discusses the need for integrity to be promoted and preserved with respect to data and systems. It recognizes that this need exists for military, public, private, and commercial organizations who depend on the integrity of their systems and their data in automated information processing, process control, and embedded computing applications. Further, it shows that this need has been recognized since the early days of computer systems development. This latter point is important in that often the argument is made that we have had no worked examples of integrity and that we need to conduct a significant amount of research before any criteria are written. This paper tries to add some balance to that argument.

The paper discusses the difficulty of trying to provide a single definition for the term integrity as it applies to data and systems. We conclude that a single definition is probably not possible and, indeed, not needed. An operational definition that encompasses various views of the issue seems more appropriate. We offer such an alternative so that progress beyond definitional aspects can be made. Our framework, or operational definition, provides a means to address both data and systems integrity and to gain an understanding of important principles that underlie integrity. It provides a context for examining integrity preserving mechanisms and for understanding the integrity elements that need to be included in system security policies. However, this study is only a beginning and remains incomplete in terms of fully addressing the topic.

The framework provides foundational material to continue the efforts toward developing criteria for building products which preserve and promote data and systems integrity. For some aspects, we conclude that there is sufficient understanding to write specific criteria, but for other aspects of such criteria, more experience, research, debate, and proofs of concepts will be needed. We believe that this partial knowledge should not delay the writing of criteria. It is the idea of concurrently pursuing both criteria and criteria-enabling research that we believe is key to making the rapid advances necessary to meet the recognized needs for integrity.

We recognize the need to establish a means to make the criteria, and thus the systems, evolvable with respect to integrity protection. Establishing this means may require more participation by systems vendors in the evolutionary development of integrity criteria than there was in the development of confidentiality criteria. The key here is to understand what is involved in designing systems for evolution so that criteria do not unnecessarily stifle new system designs or new concepts for preserving or promoting integrity.

We recommend that a criteria development study be undertaken to extend and apply the framework that has been developed in this paper. The criteria study should be conducted in parallel with protocol and mechanism demonstration/validation studies. This effort should interact with these two areas in receiving and providing direction. One major part of the criteria study should be form, a second part should be scope and specific content, a third part should address the evolution of criteria, and a final part should address the linkages of product criteria to certification and accreditation of systems by using authorities.

References

[1] Mayfield, T., J.M. Boone, S.R. Welke. 1991. *Integrity-Oriented Control Objectives: Proposed Revisions to the Trusted Computer Systems Evaluation Criteria (TCSEC), DoD 5200.28-STD*. Alexandria, VA: Institute for Defense Analyses. IDA Document D-967.

[2] Department of Defense. 1985. *DoD Trusted Computer System Evaluation Criteria*. DoD 5200.28-STD. Washington, DC: U.S. Government Printing Office.

[3] Webster's Ninth New Collegiate Dictionary. 1988. Springfield, MA: Merriam-Webster, Inc.

[4] Bonyun, David A. 1989. On the Adequacy of the Clark-Wilson Definition of Integrity. In *Report of the Invitational Workshop on Data Integrity, January 25-27, 1989, Gaithersburg, Maryland,* B.5-B.5-9. Gaithersburg, MD: National Institute of Standards and Technology.

A FRAMEWORK FOR DEVELOPING ACCREDITABLE MLS AIS

R. K. Bauer, J. Sachs, M. Weidner and W. Wilson

Arca Systems, Inc.
2841 Junction Avenue, Suite 201
San Jose, CA 95134-1921
(408) 434-6633

Abstract

Multilevel Security (MLS) is an integral requirement of many of our defense systems. Building a system to meet these requirements while still meeting stringent operational needs is quite challenging if not overwhelming. This paper highlights the tasks associated with certifying and accrediting a system to meet the security and operational needs of the end-user, then proposes a framework for integrating these tasks into the development process.

1. Overview

The ultimate objective of any Automated Information System (AIS) development or integration effort is to be accredited for operational use. To achieve this objective, the system must provide a satisfactory blend of security disciplines while accomplishing the intended mission.

Recent efforts integrating security into the development and acquisition process described in DOD-STD-2167A have focused attention on the TCSEC trust requirements of the TCB [6,7,11]. While this is a necessary condition for secure MLS operation, it is not sufficient. The fundamental premise of this paper is that prior efforts, while taking significant strides toward making trusted systems ubiquitous in all defense systems, have not gone far enough to ensure they will be operationally secure.

Operational security is often described as a chain comprised of links each of which represents a different security discipline (COMPUSEC, COMSEC, personnel security, administrative security, etc). This requires a balanced approach to allocating security requirements to each of the disciplines since the chain is only as strong as the weakest link. This collective set of requirements is the principal concern of the security certification efforts. Certification and MLS AIS development must be closely interrelated in order to achieve an accreditable system meeting its operational requirements. Key objectives of the development process and its products necessary to enforce this interrelationship are the ability to:

1) support consideration of mission requirements and security requirements prior to allocating requirements to trusted mechanisms.
2) support trade-offs between security disciplines and between overall security versus mission requirements.
3) address structure of complex integrated systems using newly developed and COTS components.

This paper presents background on the specific security tasks which must be performed and reviewed in support of certification (assessment of the overall security posture of a system in its intended operational context) and accreditation (the approval for operational use), and proposes a framework for developing Multilevel Secure Automated Information Systems (MLS AISs) meeting these objectives.

2. Certifying and Accrediting AISs

Accreditation is the step which ultimately determines whether an MLS AIS can be used to meet operational needs with acceptable risk. Although this step occurs at the boundary between development and operation, we discuss it first because it defines objectives for the earlier development and certification tasks. Accreditation is the step which determines that a system is *secure,* or more accurately, *secure enough* given the fact that no system affords absolute security. The determination of what is secure enough is made in the light of operational mission requirements, sensitivity of data, and residual risk (remaining threats and vulnerabilities) of the system in the operational environment. This decision uses the certifier's assessment of the trustworthiness of the system based on thorough review and analysis of the features and assurance the integrator has provided to make the system trusted. These words go beyond just the requirements in the TCSEC to embrace all security disciplines including those addressing personnel, physical, procedural, communications, and emanations security requirements. The integrator's assertion that the system is trusted and the certifier's assessment of the degree of trustworthiness must cover all aspects of the system's adherence to its System Security Policy.

2.1 Accreditation

The Designated Approving Authority (DAA) is typically the individual responsible for the creation and maintenance of the information resources or the execution of the mission. The DAA determines the acceptable level of risk while balancing the security of the AIS against the operational benefit of meeting the system's mission. Government policies and directives mandate protection features for each of the security disciplines based upon the information types processed and the mission accomplished. An analysis of the adequacy with which these requirements are met provides the evidence that supports the DAA's accreditation decision.

Accreditation considers the relationship between the system's trustworthiness and its operational environment. Important operational and environmental considerations include:

- Range of data processed (*e.g.*, Unclassified through Top Secret)
- User trustworthiness (e.g., clearances)
- Intended mode of operation (*e.g.*, Dedicated, System High, Multilevel)
- Location of the operation (Inside a command center or in a commercial office building?)
- The owner of the information
- What is the mission and the operational concept

The DAA considers both residual risk and operational requirements in determining if the system will be allowed to operate. The DAA decides if the system:

- May operate as planned.
- May operate if specified changes are made verified prior to operation.
- May begin operation as planned on the condition that specified changes are made within some period after initial operation.
- Will not be allowed to operate.

Required changes may affect the system design or implementation, the way the system is operated, or the environment in which the system is operated.

The most intensive DAA involvement occurs at the end of the system development process when the final review is made to determine operational suitability. However, early DAA involvement is important, specifically with respect to the Security Concept of Operations and the intended operational environment. This allows tradeoffs to be made in a manner which adequately minimizes risk while maximizing operational flexibility. The DAA also reviews the system at regular intervals (typically 3-5 years) and after major system changes. Major system changes include altering the underlying security policy, changing the threats the system was designed to counter, modifying or exchanging the components enforcing the policy, or accumulated changes which may impact security enforcement. It is important to provide information to facilitate these reviews so that the DAA can make a sound and expeditious decision. Clear policy and design documentation and rigorous configuration control are needed to support these reviews. Careful analysis of just what each component is trusted to do is essential to the efficient review of the impact of changes to the system as a whole.

2.2 Modes of Operation

Accreditation of an AIS allows it to process data in a specific mode of operation. Modes of operation are defined in DoD 5200.28. The reliance on system enforced security controls varies widely among the various modes of operation. At one extreme is *dedicated mode* in which all users are cleared for all data on the system and have a need-to-know for all data. While accountability may be required in order to determine which users have accessed which data, the system is not counted on to enforce an access control policy restricting which data users can access. Accordingly, the security features required of the system and the degree of assurance required for those features is least in this mode of operation. In TCSEC terms, it is possible that a D system might suffice for dedicated mode although a C2 system would be more appropriate even in this environment because of the accountability it provides.

In *system high* operation all users are cleared for all data but may not possess a need-to-know. The system is not relied on to control access by users to data based on classification, but it does need to provide discretionary controls which can be used to control access to data based on a user's need-to-know. In system high mode a C2 system is usually sufficient. However, the C2 system provides no means for associating classifications with data and this association may be required if output is to be disseminated to anyone not cleared for the data on the system.

In *controlled mode* or *multilevel (MLS) mode*, some users do not possess a sufficient clearance or formal access authorization to access all of the data on the system. The distinction between MLS and controlled mode is the allowed size of the difference between the least cleared user and the classification of the most sensitive data. In either case the system is relied on to control access to data based on user clearances and data classification. This means the system must implement a mandatory access control (MAC) policy. In the TCSEC, MAC enforcement is first required at the B1 level. The driving force for introducing requirements of systems above the B1 level is the need for greater assurance than that provided by a system developed to meet B1 level requirements. Additional security requirements not considered in the TCSEC may be appropriate to meet the operational site's needs in terms of data integrity and availability. (Note: in the intelligence community, *Compartmented Mode* is used where data from multiple compartments is processed on the system and not all users are authorized access for all the compartments).

2.3 Certification

Certification assesses the operational risk of a system. The certification must verify and report on the environmental factors (e.g., physical and personnel security) and determine the trustworthiness of the system. The trustworthiness of a system can be viewed in terms of the security features provided by the system and the degree of assurance that those features are properly designed, implemented and integrated. Since no useful system can provide absolute security, it is necessary to make intelligent tradeoffs between alternative designs and implementations that accurately reflect the security and functionality issues associated with these tradeoffs. This requires the development of documents which clearly and precisely describe the security policy, the system design, and the interaction of the system enforced security features with the operational environment.

It is essential that the relevant documents be stated in a form which is as accessible as possible to developers, certifiers, users and the DAA. Only if all parties understand the issues involved in the tradeoffs between security and functionality is an informed decision possible. While this may seem obvious, experience has shown that considerable care needs to be taken to make sometimes arcane INFOSEC issues understandable to those not familiar with the technology and vocabulary [4,8]. Clarity of policy, requirements, and design documentation is especially crucial when one considers the large number of parties which may participate in this process and their need to share a common understanding and terminology. Interested parties may include security advisors such as MITRE, Aerospace Corporation and NSA and organizations responsible for external interfaces such as DIA, DCA, and NSA.

The certification personnel should be involved in the early stages of system development. Evidence regarding the system's ability to meet the security requirements should be presented to the certifiers in a top-down fashion (system-wide issues followed by subsystem issues followed by component issues) during system development. Thus, feedback regarding the more abstract system design can provide guidance when making more detailed subsystem and component design decisions. Once the system is complete, a bottom-up evaluation of the system should be performed so that the certifiers can use the evidence from lower-level evaluations in their analysis of higher-level subsystems and of the entire system.

The certifiers review evidence provided by the integrator supporting the claim that the system is trusted and evidence produced by independent verification and validation activities. For a trusted system composed of integrated trusted products, certification evidence for the system as a whole will typically depend on an evaluation of certification evidence for the subsystems and components. The NCSC's CSC-STD-003-85, *Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments* provides some help in selecting the appropriate class TCB for a given application environment, but there is no latitude in constraining the operational environment. It assumes worst case operational risk environment. Landwehr and Lubbes developed an approach to use other operational factors [9] to better refine the risk index of the environments guideline by reducing the risk of exploitation in the operational environment, resulting in reducing the trust requirements of the mechanism. Neither approach went far enough in considering the impact of the operational environment, and as such is inadequate to cover the certification and accreditation of large integrated systems consisting of COMSEC and COMPUSEC components with differing levels of trustworthiness in a variety of environments.

2.4 Security Mechanisms

In an MLS AIS there will be many required security mechanisms. These services will be drawn from COMPUSEC, COMSEC, and TEMPEST. The certifiers need a vehicle to determine that the right set of security services have been provided for the operational environment. The Security Policy Statement identifies basic requirements which must be met. However, it is usually possible to meet these requirements with more reliance on environmental controls or more reliance on system enforced controls. The document which relates the system enforced controls to its environment is the Security Concept of Operations. It allocates the security requirements between TCB features and environmental controls and identifies the interrelationships between the TCB and the environmental control measures. This is the first document which explicitly identifies the security features which the system will provide. Information on how these features will work and precisely what controls are enforced is provided by more detailed security documentation such as the Descriptive Top Level Specification.

Potentially mechanisms include the following: confidentiality, accountability, data integrity, and resource availability.. Confidentiality covers Mandatory Access Control, Discretionary Access Control, and encryption. Accountability requires identification of individuals, authentication that the individual is who s/he claims to be, and audit of the user's security relevant actions. It is interesting to note that while a product evaluated against the TCSEC or TNI may come with assurance of access control and accountability features by virtue of its evaluation, it is unlikely to have been evaluated for integrity and availability features. Furthermore, integrity and availability of one component may be significant for system wide confidentiality or accountability if the component is used to store audit data or data on which access decisions are based.

2.5 Security Assurance

Trustworthiness cannot be established by emphatic assertion on the part of the developers. The integrator must provide evidence that the system is trusted. In the case of MLS AISs this evidence is reviewed by the certification team. This review, along with independent testing and analysis, determines the trustworthiness of the system.

Assurance that the system meets its security requirements must be built in as the system is developed. It is more difficult and often impossible to gain the degree of assurance required for a trusted system by after the fact testing and analysis. Testing and a variety of analysis techniques during development and integration are an essential part of gaining the required assurance. The development process must be structured so that designers and implementors are aware of system security requirements and their implications for the design and implementation tasks. The design and implementation must be structured to support analysis of the adherence of the implementation to the system security requirements. This means the design and implementation must be understandable to certifiers. Assurance evidence comes in four forms: structured design, structured development process, testing, and analysis.

Structured design supports the analysis of security requirements adherence. Structured design starts with a carefully conceived security architecture. This architecture, which allocates the system security requirements to the subsystems responsible for the enforcement of the requirements, may be presented as part of the System Security Top Level Specification or as a separate document. An effective security architecture can limit the security responsibility of subsystems and ultimately the components used to implement them. This is an application of the principle of least privilege. Use of least privilege allows certifiers to focus their review on the security critical portions of the design and implementation and to further concentrate their review on the potential abuse of particular privileges. This principle should be followed throughout the design and implementation to the largest extent possible consistent with performance and functionality requirements. Since extensive use of least privilege will certainly impact performance, and is likely to impact usage flexibility, this tradeoff must be made with skill and care.

Structured trusted development has two facets. First, security requirements must be articulated and made available to designers and implementors in a manner which facilitates their use. This requires security analysis and documentation to be closely intertwined with the system development. Security requirements need to be flowed down to more detailed design levels. The implementor of any portion of the system must be able to understand the system security requirements for the task at hand. Second control of what components, software and hardware, are introduced into the final MLS AIS, must be applied throughout the development process. Moreover, configuration control must apply to all design documentation and security documentation as well as hardware and software.

The effectiveness of testing can only be as great as the knowledge of the requirements against which to test. This emphasizes the need for an effective flow down of security requirements to subsystems and components in order to facilitate security testing at these levels. In the case of components which are evaluated products, the requirements for the component need to be reviewed against the evaluated features in order to determine the applicability of evaluation testing. If additional features are counted on to enforce system security it may be necessary to perform additional testing on the product.

With today's operational AISs testing can never be exhaustive. Also security requirements tend to be negative requirements (i.e., the system never allows certain kinds of unacceptable behavior). For these reasons security testing must be supplemented by security analysis techniques to gain assurance in the correct design and/or implementation of the security features. These techniques include: security policy models, security top level specifications, verification, covert channel analysis, security fault analysis (SFA), and penetration testing.

Security Policy Models give a precise statement of the security policy requirements enforced by the Trusted Computing Base (TCB) and the types of operations provided by the TCB. Models may be developed informally or formally, with the greater precision afforded by formal, mathematically rigorous models required for systems deployed in riskier environments.

Security Top Level Specifications provide insights into how security mechanisms work, although they do not include most implementation details. An informal *Descriptive Top Level Specification* (DTLS) describes not only the functionality provided by the TCB, but also the mechanisms used to make the TCB tamperproof and which guarantee that the TCB controls all accesses by subjects to objects. For more highly trusted systems, a *Formal Top Level Specification* (FTLS) is required. This is required in addition to, not instead of, the DTLS because formal specification languages do not support the specification of some important aspects of TCB behavior such as the interfaces between the TCB software and the hardware described in the DTLS.

Verification compares different descriptions of system behavior to show that the more concrete description satisfies all of the requirements of the more abstract description, for example, one can verify that an FTLS meets the requirements of the Security Policy Model. If both descriptions are formally presented, a formal verification, using mathematical proof, can be done. Formal verification is only practical at the design level (e.g., Model-FTLS verification) because the amount of detail at more concrete design levels and in the implementation quickly make formal verification using current state-of-the-art techniques for systems of even moderate size intractable.

Covert Channel Analysis is a technique for finding information flows contrary to the System Security Policy. Covert channels exist due to the possibility that the modulation of shared resources by one subject (or process) can be detected by another, even if the System Security Policy would normally prohibit communication between the two.

Security Fault Analysis is a technique familiar to the COMSEC community. Whereas the COMPUSEC community has put a large emphasis on software verification, SFA has focussed on analysis of hardware and the effect of faults on the security of the component. This was originally done when the complexity of devices made analysis down to the gate level practical. These techniques are now applied to more complex hardware bases. In addition to the continued need to apply SFA to critical hardware components, the principles of SFA provide lessons to COMPUSEC design and development such as the significance of single points of failure.

Penetration Testing assesses the strength and effectiveness of security features by means of an attempt to circumvent those features. The penetration testers analyze the system design and implementation for potential flaws and then attempt to utilize those flaws to penetrate the system. The results of penetration testing are only meaningful if it is carried out by experienced individuals.

2.6 System and Security Evolution

The discussion above was primarily from the point of view of a newly developed MLS AIS. Actually, most MLS systems are typically based on existing systems. Even when a system is developed from scratch with MLS as an objective, it is likely that the need to promptly address user requirements will necessitate an initial operating capability (IOC) with capabilities which will evolve as new technology becomes available. The system may also have to evolve because of changes in the environment which reduce or increase the reliance on system enforced security. Such development and integration efforts in the past have used a modified waterfall development model to provide some form of iterative development cycle [6,11]. New development models, for example the Spiral Model developed by Boehm [15], are being investigated for their contribution to the development of trusted systems [5].

The evolution of systems needs to be viewed from the point of view of its impact on system functionality, performance, and trustworthiness. New commercial-of-the-shelf (COTS) products are not an end in themselves. Rather they are useful in so far as they make it possible to provide increased functionality, better performance, or a higher degree of trustworthiness.

The certification team needs to be involved in the consideration of proposed improvements in order to determine the security ramifications of the changes. Just as with the original certification, the certifiers will need to provide information to the DAA which allows the DAA to determine if the level of risk is acceptable. If the certifiers are involved in the early consideration of proposed changes they can provide input as to whether the change will unacceptably affect security. If that is the case, the certifiers can propose alternative changes or recommend no change be made. If this analysis is done before work on the change has proceeded very far, wasted effort can be averted. The DAA must accredit the altered system. Coordination with the DAA should define the level of change requiring reevaluation and whether interim operation of the altered system may take place before final DAA approval.

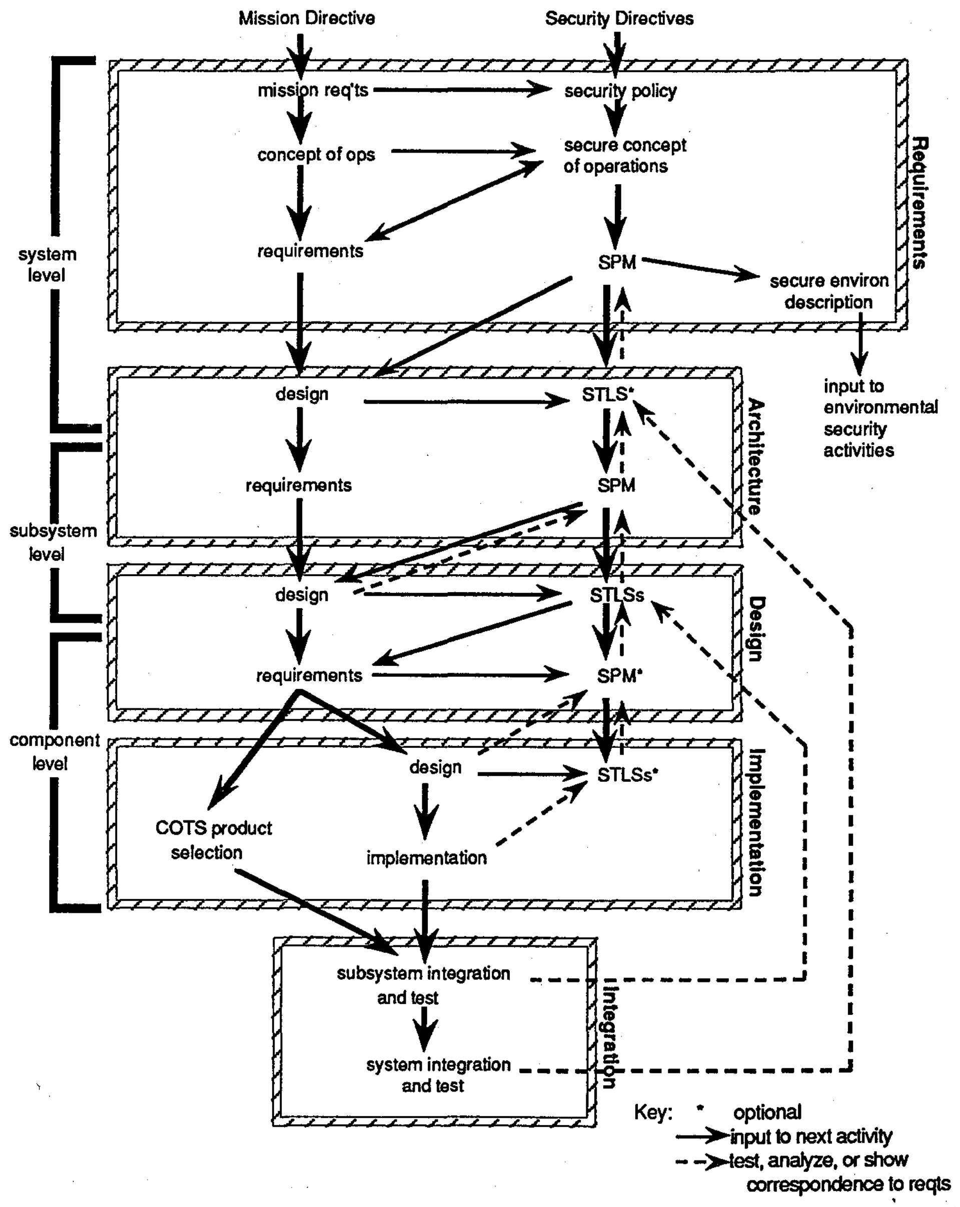


Figure 1: Steps in MLS Development

3. MLS AIS Development

The discussion to this point has highlighted the various tasks required to support certification and accreditation. Failure to integrate security requirements and the attendant deliverables into the development process and products has historically resulted in systems that were either operationally deficient, unsecure or both [4,8]. Since security cannot be retrofitted, these tasks must be carefully integrated into the system development process. The security requirements must be clearly understood by all parties and appropriate requirements reflected throughout the design and development. It must also support informed tradeoffs between security, performance and functionality for alternative design and implementation approaches.

With these goals in mind we present the framework in Figure 1 as an approach to how particular documents and activities are related to the overall development and certification process. For clarity, the security tasks are called out from the standard development tasks, but the security tasks must be executed in close collaboration with the development tasks or fully integrated with the development process and products. The approach allows for separate security deliverables for COTS trusted products with existing security policy models, top level specifications, etc. The arrows in Figure 1 represent primary inputs. Later tasks will often identify required changes in the results of earlier tasks providing necessary feedback, for example, the development of a Top Level Specification may reveal deficiencies which must be corrected in the system design. Certainly the form these various activities and documents take will vary, especially when the process is used for an evolving system. The figure identifies some items which are optional depending on the complexity of the system and its subsystems. However, it is important that security tasks are performed which allow the tracking of security requirements through all development steps and that these tasks support intelligent and timely tradeoffs between operational flexibility, life cycle costs, performance, and security.

3.1 Requirements

Requirements are driven by mission directives and security directives. Applicable directives and their implications for the particular system under development are captured in the System Security Policy Statement. The System Security Policy Statement specifies the security requirements the system, in conjunction with environmental security controls, must enforce. The System Security Policy must be stated in the context of the mission requirements.

The system security policy must be complementary to the administrative, procedural, physical, and personnel controls present in or anticipated for the operational environment. The document which describes the interaction of system enforced controls and the system environment is the Security Concept of Operations. Both the System Security Policy and the Security Concept of Operations define the requirements for system security features. The Security Concept of Operations is an important document for supporting the intelligent determination of tradeoffs between security controls in the environment and system enforced controls. A Security Concept of Operations can be written to describe phases through which the MLS system may evolve. The Security Concept of Operations must be consistent with the System Concept of Operations. Likewise, the System Concept of Operations must reflect the System Security Policy.

Since the Security Concept of Operations and the System Security Policy define security requirements, they must be used by the developers to integrate security into the functional requirements. The System Security Concept of Operations yields both environmental requirements and system security requirements. This is noted in Figure 1, although the system secure environment description is likely to be a portion of the System Security Concept of Operations rather than a separate document. On the other hand, the System Security Policy Model will typically be a stand-alone document which describes the specific properties which must be enforced by the system TCB and the types of operations supported by the system TCB.

3.2 System Architecture

After the system's functional and security requirements have been established, a system architecture must be developed which defines subsystems and the functional and security requirements on those subsystems. The first step in this process is the development of system functional design. This has to reflect the system functional requirements and the security requirements as described in the Security Policy Model. Depending on the complexity of the system it may be desirable to have a Security Top Level Specification, either a DTLS or a DTLS and FTLS, based on the level of trustworthiness required. However, since security requirements on subsystems are reflected in Subsystem Security Policy Models, it may be possible to incorporate sufficient information about subsystem interaction in the System Security Policy Model and in that way omit an explicit Security Top Level Specification. The subsystem requirements drawn from the system functional design complete the system architecture phase. The Subsystem Security Policy Models must reflect the security requirements allocated to that subsystem and the impact of the functional requirements identified in the subsystem's requirements statement.

3.3 System Design

In this phase subsystem TLSs are developed to describe the security features implemented in the subsystems. Because component Security Policy Models will not always be available, the subsystem TLSs will be relied on to describe how security features work in each subsystem and how components interact to implement those security features. Also, the role of the component Security Model may be replaced by other documents such as the Software Requirements Specification for COMSEC components. The component requirements must reflect both the functional requirements flowed down in the subsystem designs and the security requirements in the subsystem Security Top Level Specification. These component requirements form the basis for selection of COTS products (whether COMSEC or COMPUSEC) or the design and implementation of newly developed components.

3.4 System Implementation

System implementation is accomplished through the design and implementation of the newly developed components which comprise the system and the selection of COTS products in the case of components for which suitable products exist. Security specifications for components detail the security aspects of the component design. Depending on the type of component, the nature of the component security specification may vary significantly. For a complex trusted component, whether newly developed or a COTS product, the specification may take the form of a traditional Security Top Level Specification. For a COMSEC component, the STLS may take the form of a Software Program Specification. In the case of particularly simple components, the component STLS may be omitted.

3.5 Integration and Test

The steps identified in the framework provide the basis for security test and evaluation. Figure 1 shows where testing and analysis techniques can be used to ensure that security design and requirements have been accurately followed. Security tests on subsystems can be performed using test cases developed from the Security Top Level Specifications for the subsystems. For those components where Security Policy Models and/or Security Top Level Specifications have been developed, these documents can be used to generate test cases for component testing. Otherwise, the Subsystem Security Top Level Specification will have to be relied upon to provide sufficient detail on the required security behavior of the component to serve as a starting point for test case generation. Security tests for the integrated system can be performed based on test cases derived from the System Security Top Level Specification, if one has been developed, or directly from the Informal Security Policy Model if it is sufficiently detailed.

3.6 Security Analysis

Section 2.5 described techniques which can be used to perform security analysis at various points in the system development. Verification can be used to demonstrate that the System Security Top Level Specification meets the requirements of the Security Policy Model and that the subsystem Security Policy Models meet the requirements of the System Security Top Level Specification. Alternatively, the Subsystem Security Policy Models can be shown directly to meet the requirements of the System Security Policy Model. For each subsystem, verification can be used to justify that the design reflected in the Subsystem Security Top Level Specification is consistent with the Subsystem Security Policy Model. Where component models and security specifications exist, the verification can be carried down to that level. Otherwise, testing and review of correspondence of the component implementation to the requirements of the Subsystem Security Top Level Specification can be used to show that the component satisfies its specified security requirements.

Penetration testers will use all of the documentation and specifications generated in this process to determine potential faults to exploit. The penetration tester must eventually test these potential faults by attempting to penetrate the integrated system. However, some faults may be dependent solely on the functionality of a particular component or subsystem. These faults can be tested as soon as the component or subsystem is available without waiting for integration. This early feedback can support penetration testing which is extensive enough to provide reasonable assurance, and the need to deploy the system promptly.

3.7 Tradeoffs

The framework described above allows security and functional requirements to be considered at all stages of system development so that intelligent tradeoffs can be made. The first tradeoffs are made in developing a System Security Policy which is sufficiently stringent to meet the requirements of relevant directives and yet flexible enough to allow mission requirements to be met. The next tradeoff is between system enforced and environmentally enforced security. This tradeoff is made as the Security Concept of Operations is developed, reviewed, and updated. In the system architecture phase the system functional design is the vehicle for allocating requirements to subsystems. From the security point of view this allocation should be made in such a way as to minimize and simplify the TCB. However, these considerations need to be considered in the context of their effects on performance and flexibility. In the implementation phase, tradeoffs between least privilege and performance will again need to be made as the subsystem designs are developed. At this stage an important factor in that tradeoff will be the availability of evaluated products which can provide some of the security enforcement.

Finally, it should be noted that this is necessarily an iterative process, for example, as subsystem designs are developed it may become clear that a reallocation of requirements among subsystems would enhance security, performance, functionality or some combination of these. Also, as new COTS products become available, an altered subsystem design or even system functional design may be appropriate. It is important to enforce configuration control on this process so that even with iterations the set of accepted documents, specifications, and implementations are consistent.

3.8 Operational Documentation

One of the important outputs of the MLS system development process is the documentation which tells privileged users and other users how to interact with the system and maintain security. Improperly used security controls can be just as vulnerable as insufficient or improperly implemented controls. Two key documents which tell users how to properly use and maintain the security features are the Trusted Facility Manual (TFM) and Security Features User's Guide (SFUG). The TFM describes the functions available to privileged users such as the system administrator and system security officer as well as what these users must do to properly initialize and maintain the system, and securely recover from system failures. The SFUG explains to general users what security controls are enforced and what the user's role is in conforming to the security policy. Proper training of all users on their security responsibilities and more extensive training for privileged users on how to keep the system secure are essential for the operational system to be run securely. Clearly written, comprehensive documentation plays a central role in making sure the users understand their security responsibilities.

4.0 Conclusion

This integrated approach to system development and security engineering allows effective tradeoffs between system security controls and operational requirements to minimize the total cost of development, operation, and maintenance. It ensures that the broader set of security requirements, and not just trust requirements, are adequately considered throughout the development process. Finally, it supports the integration of complex systems comprised of trusted and untrusted COTS products, and newly developed components. This MLS AIS development approach provides the basis for successful certification and accreditation of the fielded operational system.

5.0 References

[1] Department of Defense, Security Requirements for Automated Information Systems DoD 5200.28, March 1988.

[2] National Computer Security Center, Computer Security Requirements -- Guidance for Applying the DoD TCSEC in Specific Environments, CSC-STD-003-85, 25 June 1985.

[3] National Computer Security Center, Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, December 1985.

[4] D. J. Bodeau and M. J. Reece, "A Multilevel-Mode System for Space Applications: Lessons Learned," in *Proceedings of the Sixth Computer Security Applications Conference*, IEEE Society Press, December 1990.

[5] A. Marmor-Squires, B. Danner, J. McHugh, L. Nagy, D. Sterne, M. Branstad, and P. Rougeau. "A Risk-Driven Process Model for the Development of Trusted Systems," in *Proceedings of the Fifth Computer Security Applications Conference,* IEEE Society Press, December 1989.

[6] T. C. V. Benzel, "Developing Trusted Systems Using DOD-STD-2167A," in *Proceedings of the Fifth Computer Security Applications Conference*, IEEE Society Press, December 1989.

[7] S. D. Crocker and E. J. Siarkiewicz, "Software Methodology for Development of a Trusted BMS: Identification of Critical Problems," in *Proceedings of the Fifth Computer Security Applications Conference,* IEEE Society Press, December 1989.

[8] C. R. Pierce, "Experiences in Acquiring and Developing Secure Communications-Computer Systems," in *Proceedings of the Thirteenth National Computer Security Conference*, National Computer Security Center, October 1990.

[9] C. E. Landwehr and H. O. Lubbes, *An Approach to Determining Computer Security Requirements for Navy Systems*, Technical Report NRL 8897, Naval Research Laboratory, May 1985.

[10] A. C. Hoheb, "Integrating Computer Security and Software Safety in the Life Cycle of Air Force Systems," in *Proceedings of the Thirteenth National Computer Security Conference*, National Computer Security Center, October 1990.

[11] T. C. V. Benzel, "Integrating Security Requirements and Software Development Standards," in *Proceedings of the Twelfth National Computer Security Conference*, National Computer Security Center, October 1989.

[12] W. Norvell, "Integration of Security into the Acquisition Life Cycle," in *Proceedings of the Twelfth National Computer Security Conference*, National Computer Security Center, October 1989.

[13] F. Tompkins and R. Rice, "Integrating Security Activities into the Software Development Life Cycle and the Software Quality Assurance Process," *Computers and Security,* Vol. 5, No. 3, September 1986.

[14] D. E. Bell, "Working Towards A1," in Proceedings of Seventh DOD/NBS Computer Security Conference, DOD Computer Security Center, September 1984

[15] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, May 1988.

[16] M. S. Deutsch and R. R. Willis, *Software Quality Engineering*, Prentice-Hall, 1988.

GENERALIZED FRAMEWORK FOR ACCESS CONTROL: TOWARDS PROTOTYPING THE ORGCON POLICY[1]

Marshall Abrams* Jody Heaney* Osborne King* Leonard LaPadula+ Manette Lazear* Ingrid Olson*

The MITRE Corportion *7525 Colshire Dr., McLean, VA 22102
+Burlington Rd., Bedford, MA 01730

1 INTRODUCTION

The Generalized Framework for Access Control (GFAC) was introduced in [1, 4] as a framework for studying and constructing access control policies in Automated Information Systems (AISs). This paper discusses a prototyping effort that uses the GFAC concepts. Further, it describes a security policy and the experience gained through implementing a prototype based on that policy.

GFAC asserts that all access control policies can be expressed as *rules* specified in terms of *attributes* and other information controlled by *authorities*. All policies can be expressed within this framework, including policies conventionally implemented through trusted processes and privilege mechanisms. The GFAC concepts include four factors representing dimensions of choice and constraints to the designer of a trusted AIS:

- Access Control Information (ACI) – Characteristics or properties of subjects and objects. ACI names are used in specifying the rules of the system; their values are used by the access control rules.
- Access Control Context (ACC) – Additional information, such as time of day, used in access control decision making.
- Access Control Authorities (ACA) – Authorized agents who specify ACI, ACC, and rules.
- Access Control Rules (ACR) – The set of formal expressions of policy for adjudicating requests by subjects for access to objects.

1.1 Markings

Within the DOD/intelligence community, numerous dissemination/handling restrictions and markings are applied to the manual handling of classified documents. Examples include NOFORN (Not Releasable to Foreign Nationals), ORCON (Dissemination and Extraction of Information Controlled by Originator), and REL XX (Authorized for Release to (name of country(ies)/international organization), which are defined in DCID 1/7 [2]. Williams and Day [8], and also Graubart [3], provide excellent discussions of the complexities of such markings for classified documents, and the inadequacies of current automated systems in handling them.

Such restrictive control markings are examples of a class of existing access control policies that limit the dissemination of information beyond the traditional Mandatory Access Controls (MAC) and Discretionary Access Controls (DAC) specified in the Trusted Computer System Evaluation Criteria (TCSEC) [5]. MAC and DAC have almost become synonymous with access control in automated systems, when in practice there are many other policies in existence in the paper world that are reasonable candidates for automation. Although MAC in particular, and DAC to some degree, are useful and reasonable policies for some environments, support for additional policies in automated systems is needed. The GFAC effort is attempting to demonstrate that a more general, useful model of access control is feasible and necessary to support the many access control policies.

In the DOD/intelligence community, other policies must be satisfied in addition to MAC (i.e., in addition to having the appropriate security clearance, the user must also satisfy the access rules of the additional policy). In the unclassified world, such policies may be implemented through non-disclosure agreements and contractual limitations on information disclosure. MAC *may* not be a requirement in conjunction with other non-DOD policies. For example, the Bureau of Labor Statistics uses RELEASABLE AT <time,date> to safeguard unemployment figures. This information is highly protected until <time,date> when it is widely distributed.

Using GFAC, appropriate markings and other supporting information needed to make access control decisions to implement such restrictive control markings can easily be included as subject/object ACI or additional ACC

[1] This work was supported in part by the MITRE Corporation as MITRE-Sponsored Research and in part by the U.S. Army as Mission-Oriented Investigation and Experimentation under contract DAAB07-91-C-N751. Technical direction for the research was provided by the National Security Agency.

information. Development of the necessary access controls is theoretically straightforward using GFAC. Note that the strength or universal applicability of access control rules is independent of the information on which the rules base their decisions. Thus, the implementation of a marking policy can be just as strong and pervasive in a trusted system as the implementation of a traditional MAC policy.

1.2 Prototyping based on the GFAC Concepts

To provide a tangible proof-of-concept, we are developing a prototype for one of the additional policies noted above that was, in turn, expressed using the GFAC concepts. ORCON is the most restrictive policy defined in DCID 1/7 and, therefore, was selected as the basis for an automation policy. Development of an ORCON-like policy has been instructive; this paper is intended to help share some of the experiences. The following sections discuss the ORGCON policy, the ORCON-like policy that forms the basis of the prototype, and numerous prototype issues, design decisions, results, and lessons learned. There are some characteristics of ORCON, such as special instructions relating to incorporation or retention period, that were not included in the ORGCON policy. The term ORGCON, instead of ORCON, is used so that a precise AIS policy can be implemented without usurping the Government's definition(s) of ORCON.

2 ORCON POLICY

The Organization Controlled (ORGCON) policy (described in Section 3) was developed as a practical example of using the GFAC concepts. The ORGCON policy is a policy for AISs that builds upon the ORCON ("Dissemination and extraction of information controlled by originator") dissemination control on paper documents. In choosing to develop ORGCON and prototype based on this policy, we are attempting to transfer a well-established policy from the control of paper documents to the control of information in an AIS. For completeness, this section provides a high-level description of the ORCON policy that the ORGCON policy is based on.

2.1 The ORCON Dissemination Control

ORCON is only one of a number of restrictive control markings defined in DCID 1/7 applied in the dissemination and use of intelligence information and related materials. These markings represent handling policies that limit the authority of recipients of the information to use or transmit it. ORCON requires the permission of the originator to distribute information beyond the original receivers designated by the originator. For the purposes of this paper, the following extract from DCID 1/7 defines the ORCON marking:

> This marking is used, with a security classification, to enable a continuing knowledge and supervision by the originator of the use made of the information involved... Information bearing this marking may not be disseminated beyond the headquarters elements of the recipient organizations and may not be incorporated in whole or in part into other reports or briefings without the advance permission of and under conditions specified by the originator.

2.2 The ORCON Originator and Recipient

The originator has not only the right, but the responsibility to identify and mark information as ORCON information. The originator also has the responsibility to explicitly identify what organizations will be indicated on the distribution list for the specific information.

"Originator" is not defined in DCID 1/7. We believe that the authority to control dissemination of the information rests within an office or organization. An individual may only have dissemination authority by virtue of acting on behalf of that office or organization. This implies that the originator of ORCON information is never an individual user. In fact, the originator is always an office or organization code or some analog thereto. An individual does not own such information any more than an Air Force pilot owns an F-15. Similarly, ORCON material is never addressed and distributed to an individual. Some information may, however, be addressed to a commander only. Even this information is likely to be handled by a limited number of people in addition to the designated recipient (e.g., executive officer). The term designated recipient, in the preceding sentence, does not connote an individual, but rather the role filled by an individual (e.g., Commander-in-Chief (CINC)).

2.3 ORCON Information Dissemination

ORCON information may be received and processed in a number of different ways. Message traffic is the most common. Messages may be read from a terminal, posted on a read-board, or routed as paper-copy. Access to the data may be via remote terminal access from a terminal to a database at a central location. Information may also be received via a dedicated or special purpose system.

The internal access to and distribution of ORCON marked information depends on its form and content, as well as the number of staff with assigned responsibilities in the area related to the information. Access and distribution are also dependent on the tools available to process ORCON information.

The originating organization for ORCON information is either explicit (a message has a "from" address indicating the originating organization) or implicit (remote access to a database implies that the organization hosting the database is the originator). ORCON information is transmitted by some method (e.g., photo-copy, electronic transmission) that effectively creates a new copy of the information at the destination. Handling, retention, and destruction of ORCON information, by both the originator and recipient organizations, varies. Handling, retention, and destruction depend, in part, on other security markings on the information. Old copies of information may be destroyed after database updates. Reports containing ORCON information are more likely kept on file for a specified period of time.

3 THE ORGCON POLICY

The ORGCON policy uses the ORCON policy concepts applied to paper documents, but was developed as an AIS policy to control the dissemination of information. ORGCON is a policy for non-discretionary group-based access control. Groups are discussed further in Section 4. The primary elements of the ORGCON policy are as follows:

- ORGCON information is owned by an originating organization and ownership is not alterable.
- ORGCON information is distributed only to an identified list of recipient organizations.
- The list of authorized individuals of each recipient organization is maintained by a recipient representative.

3.1 Originator and Recipient Representative Roles

The ORGCON policy controls the ownership and dissemination of ORGCON information (information marked ORGCON). ORGCON information is owned by its originating organization. The originating organization is represented by one or more individuals acting in the role "originator representative" (ORGREP). Any individual may generate information that may eventually be designated with the ORGCON marking, but only an ORGREP can mark the information ORGCON and specify a distribution list of recipients.

Individuals acting in the role of "recipient representative" (RECREP) specify the individuals who are authorized to receive ORGCON information at the recipient organization.[2] Example recipient organizations include the headquarters staff and CINC. Note that ORGCON differs from ORCON by the introduction of the RECREP, which is believed to be a necessary and practical step. The ORGREP cannot be expected to be aware of personnel changes in the recipient organization, nor will (s)he be likely to have the privileges to redefine the membership of the recipient organizations. The originator and RECREPs are authority agents (authority component of GFAC), perhaps the Information System Security Officer (ISSO) or Security Administrator.

3.2 ORGCON and ORGCON-C Markings

Two markings are defined for the ORGCON policy. The ORGCON marking identifies an object as being under ORGCON policy control. ORGCON-C is a special marking that identifies an object as a candidate for handling under the ORGCON policy. The ORGCON-C marking identifies the object as being write-accessible only to the individual user who created it. By convention, this user is referred to as the owner of the ORGCON-C object. The owner may read, write, or delete the object. The ORGREP may read the object and is privileged to change the marking; the only authorized changes are from ORGCON-C to ORGCON. A normal progression would be for the individual owner to pass an ORGCON-C object to the ORGREP for marking as ORGCON and distribution.

3.3 Reading ORGCON Objects

An individual can only obtain read access to ORGCON information if the individual is a member of a recipient organization that is on the distribution list. This condition for read access holds for the creator of the ORGCON information and all representatives of organizations. That is, once the information is marked ORGCON, there are no exceptions to the conditions for read access. In order for an ORGREP or RECREP to be able to read an ORGCON object, they must be members of a group named on the distribution list. As a practical matter, the ORGREP role will

[2] Note that the ORCON policy identifies the headquarters element of an organization as the recipient. The ORGCON policy has been generalized and does not imply the headquarters element as the recipient.

probably be placed on the distribution list and the RECREPs will be part of each respective recipient organization. Other policies can be envisioned under varying circumstances.

3.4 Copying ORGCON Objects

In the process of distributing ORGCON information, multiple copies of the information may be generated. Many different mechanisms could be employed for distributing ORGCON objects within a recipient organization, depending on the AIS architecture employed. For the purpose of this paper, we discuss two possible architectures. The first architecture has only authorized users accessing a shared file system (e.g., a single multi-user system, a shared file server). Given this architecture, only one copy of an ORGCON object is required. The second architecture has users without access to shared file systems (e.g., separate single or multi-user systems, non-client-server workstations). These users will require individual copies. Therefore, the ORGCON policy must control the copying of ORGCON information, as well as its final disposition. The original of any ORGCON information logically resides with the originating organization.

The major points of the ORGCON copy policy are identified below and discussed in the following paragraphs.

- Only RECREPs or a daemon performing privileged system operations can copy ORGCON information for distribution to those individuals defined as recipients.
- Any recipient of an individual copy of ORGCON information can view and dispose of his/her own copy of the information.
- No individual recipient of ORGCON information can copy that information.

Many schemes for marking ORGCON objects are possible. In one scheme, an object marking has two fields, an ORGCON field and an ORGCON-copy-control field. When an object enters the ORGCON system it is marked ORGCON by the ORGREP. At this point the ORGCON-copy-control field defaults to Null. This configuration of marking automatically identifies the object containing the information as the original version. When a copy is made the ORGCON-copy-control field is filled-in. This could be done in several ways. The field could contain a copy number or some designation which identifies the recipient of the copy. This is summarized in the table 1.

Table 1. Possible Implementation of ORGCON Control Fields

STATUS	FIELD	
	ORGCON	ORGCON-Copy-Control
ORGCON original	ORGCON	Null
ORGCON copy	ORGCON	Recipient ID

The RECREP (or daemon) is privileged to copy ORGCON objects for distribution to those users defined as belonging to the recipient organization. This distribution may be performed manually, but is performed by a process (e.g., a daemon) with the privilege to make and distribute the copy, running on behalf of the RECREP. Each copy of ORGCON information created carries the distribution list for that information and an identifier for the originating organization.

Individuals in the recipient organization who are not RECREPs may not copy ORGCON information. The access of these individuals to ORGCON information is limited to reading and disposal. Each individual recipient is responsible for proper disposal of their individual copy of ORGCON information. The RECREP may delete the recipient organization copy of the ORGCON object. ORGREPs are responsible for proper disposal of the original ORGCON information.

3.5 Handling an ORGCON Object

There are several different roles associated with the ORGCON policy, and each role has different responsibilities and privileges associated with it. In the prototype, two roles are implemented: the ORGREP and the RECREP.

When an object is marked ORGCON, the ownership of the object is changed to the ORGREP, the designated authority for marking and extending access to ORGCON objects. The ORGCON policy rules specify the authority of the originator representative role and the recipient representative role relative to granting read access to ORGCON objects.

An important feature of the ORGCON policy is that the distribution list for ORGCON information is part of the object. The ORGREP is the only role responsible for creating the distribution list (DL) for an ORGCON object. The policy decision was made that once an ORGCON object is created and the distribution list attached, no changes can be made to the list of recipients. At the time of distribution, the ORGCON object should be thought of as including the DL. Consider that the aggregation of object and DL could change the hierarchical level classification. We have chosen not to implement this in the current prototype, but it is one example of why the ORGCON policy forbids changes once the DL is attached.

3.6 ORGCON Control of Access

The access control rules of the ORGCON policy are summarized in table 2.

Table 2. ORGCON Control of Access

WHEN THE REQUESTED ACTION IS:	THE FOLLOWING CONDITIONS MUST BE MET:
Mark as ORGCON-C	User is owner
Change ORGCON-C to ORGCON	User is ORGREP
Read ORGCON-C object	User is owner or ORGREP
Read ORGCON object	User belongs to a recipient organization, or daemon
Delete ORGCON object copy	User received an individual copy of the ORGCON object, or copy belongs to recipient organization and user is RECREP
Delete ORGCON object original	User is ORGREP
Copy ORGCON object	User is RECREP, or daemon
Write ORGCON-C object	null

4 ROLES/GROUPS AND DAC

4.1 Roles and Groups

To develop a prototype for the ORGCON policy, identification of several roles (i.e., equivalence classes of users) is required. Each of these equivalence classes is identified by name. The TCSEC implicitly defines groups as part of the specification of DAC as follows:.

> The enforcement mechanism ... shall allow users to specify and control sharing of those objects by named individuals or defined groups of individuals, or both...

The Trusted Network Interpretation (TNI) distinguishes between users and roles:

> Note that "users" does not include "operators," "system programmers," "technical control officers," "system security officers," and other system support personnel. They are distinct from users and are subject to the Trusted Facility Manual and the System Architecture requirements. Such individuals may change the system parameters of the network system, for example, by defining membership of a group. These individuals may also have the separate role of users.

The concept of named equivalence classes of users, however, is too important a concept to be used only with DAC. The usage has, therefore, been extended by prepending the policy name as an adjective when necessary for clarity (e.g., DAC-group, ORGCON-group). The meaning is clear: the members of this identified set of users are to be treated identically with respect to the specified policy. There may be multiple groups, each having different privileges relative to the specified policy.

Informally, a group is a collection of users that share a set of access control attributes. An individual member of the group may act with any of the access privileges authorized for the group. The composition of a group is determined by an appropriate authority, and a primary purpose of creating groups is essentially administrative convenience. However, it is important to note the support for separation of function provided by groups. A role may be viewed as a particular kind of group. The distinguishing feature of a role is the identification of unique privileges with respect to the stated policy. When a user takes on a role (usually explicitly), the user relinquishes the privileges associated with their previous role. A role is not associated with an individual user, but with a set of users (i.e., a group) authorized for the specific role. ORGCON-roles defined in this paper are summarized in Table 3.

Table 3. ORGCON-Roles

ROLE	FUNCTION
Originator representative	Marks an ORGCON-C object as ORGCON and affixes the distribution list (list of recipient organizations)
Recipient representative	Controls membership of recipient organization; may copy ORGCON object for distribution to recipients

4.2 Traditional DAC Policy

This effort has caused us to explore the nature of DAC and how it fits in the GFAC view of access control policies and their implementations. Primarily, DoD Directive 5200.28, the TCSEC, and the DAC Guide [6] have been consulted. It appears that the term DAC is used interchangeably to refer to both a set of mechanisms and a policy. The DAC policy defined by the TCSEC is referred to here as traditional DAC. Traditional DAC allows an authorized user to determine who is authorized what mode of access to an object. Nothing is stated about how the user receives authorization for specific modes. In the literature, the initial authorized user is often identified as the owner of an object, but this is not necessarily the case. For that matter, the concept of ownership is not universally defined. The DAC policy is really a special case of the principle of least privilege; that special case is need-to-know.

There are also numerous supporting policies that are NOT associated with DAC. With hindsight and the benefit of the GFAC perspective, we note that many, perhaps all, of the weaknesses attributed to traditional DAC actually identify the absence of supporting policies. GFAC provides an opportunity to experiment with the design of other identity-based policies to overcome DAC deficiencies and to meet other policy objectives. Nothing in the TCSEC prohibits the addition of these or other supporting policies to DAC. However, it is not clear if anyone has ever done so. A precedence has thereby developed defining traditional DAC.

The two major shortcomings of traditional DAC are the lack of an inheritance policy and the lack of accountability. The lack of an inheritance policy means that the mechanism only protects the container, not the information. Once the information is read from the container, there are generally no controls on what can be done with the information; there is no ability to control copies. The lack of accountability means that the DAC mechanisms are vulnerable to Trojan Horses, since programs executing on behalf of a user generally assume the privileges of the user.

5 THE PROTOTYPE

In this section, the goals of this prototyping effort are described and an overview of the System V/MLS prototyping environment is provided. Some advantages, difficulties and limitations of adding an additional policy to an existing secure system [9] are also discussed.

5.1 Prototype Goals

There are two main goals to this initial GFAC prototype effort:

1. To demonstrate a prototype based on the GFAC concepts.
2. To implement an access control policy, namely ORGCON, in addition to MAC and DAC.

To demonstrate the GFAC concepts, the prototype must satisfy the following three goals. Note that accomplishment of these goals makes it possible to implement any access control policy.

1. The prototype must provide for the creation, maintenance, and change of those ACI relevant to the particular access control policy being implemented.
2. The prototype must provide the appropriate set of rules necessary to implement the given policy.
3. The prototype must embody an explicit definition of authority with respect to the given policy, either through well-defined roles, through the rules, or in the ACI.

The second goal of the GFAC effort, to implement an additional policy, is important in order to demonstrate the feasibility of implementing policies other than MAC and DAC.

For the sake of expediency, a system that already provides a B-level MAC policy was used as the prototype base. That is, an existing TCB was modified to execute additional policies. We expected that many of the mechanisms used to implement MAC sensitivity labels might carry over to the handling of the ACI for additional policies. Portions of the TCB outside the kernel (i.e., the reference monitor implementation) were expected to be directly useful. AT&T System V/MLS [7] was selected as the host base for development of the prototype.

5.2 Prototype Environment

System V/MLS supports two types of access controls: DAC and MAC. The discretionary controls provided are identical to those controls provided by standard UNIX System V. DAC permits owner control of access to resources by other users, and is implemented via the user/group/other mechanism. Permission to read, write, and/or execute (for files) and search (for directories) may be set for each class of users (owner of the object, group associated with the object, and for others (all system users)).

In addition, System V/MLS provides mandatory controls, defining access to resources based on labels. System V/MLS controls access to resources using the current operating privilege of the user and the privilege requirements associated with a resource. Privilege is the term used to refer to the DAC group and the MAC label associated with a user or a resource. The label is the combination of a hierarchical level and zero or more categories. A privilege can be thought of as an instance of a group at different levels and categories. While an object has only one privilege associated with it, users may change their current operating privilege (i.e., the label and group associated with them). The range of labels over which a user may operate is referred to as the user's clearance. A user, however, may not necessarily be a member of all privileges defined in that range.

Files or directories may only be created in a directory that has a label identical to the user's current operating label. Once created, however, the files/directories' labels may be upgraded. Within DAC and MAC, files and directories are accessed based on the user's current operating label (i.e., the label part of the privilege). This label must dominate (for read access) or be identical to (for write access) the label of the file/directory the user is trying to access. Formal models of System V/MLS and of the ORGCON policy are in preparation.

5.3 Difficulties/Tradeoffs

While developing the prototype on an existing security system has its advantages, there are also numerous difficulties and tradeoffs in retrofitting an existing system. A major dilemma was deciding which of the following two objectives took precedence in the prototype:

1. Strict adherence to the GFAC concepts and structure which could require extensive changes or additions to the existing system base.
2. Implementation of the additional policy using capabilities of the existing system without necessarily demonstrating the GFAC concepts.

Occasionally, the effort was limited by existing structures within System V/MLS that were not alterable. Therefore, 'workarounds' were devised to implement the controls of the ORGCON policy. For example, the privilege concept in System V/MLS, the coupling of the label (i.e., hierarchical level plus any categories) and the DAC group, and the mechanism for the user's current operating privilege, restrict how a subject can access an object. These controls are strict and useful with respect to MAC and DAC. ORGCON, however, has additional controls and a different set of groups (ORGCON-groups vs. DAC-groups) that presented difficulties during incorporation into the existing structure.

Part of the difficulty was related to our attempt to strictly adhere to the GFAC concepts. It was desirable to implement ORGCON using data structures that clearly mirrored the GFAC concepts of ACI and ACC. The structures finally chosen were rationalized as practical compromises that do not violate the GFAC concepts nor the System V/MLS mechanisms.

In some cases, working on an existing secure system resulted in a less than ideal balance in terms of achieving the stated goals. GFAC provides a high-level informal model of access control in AISs (i.e., an abstraction). The restrictions of an actual system forced the sacrifice of implementation of the prototype strictly according to the GFAC concepts.

5.4 Implementing Based on the ORGCON Policy

This section discusses the actual implementation effort on the prototype to date (June 1991). Primarily, the discussion focuses on our current thinking with regard to best approaches for implementation of the controls to support the ORGCON policy. Some details and issues remain to be resolved.

5.4.1 Observations on Implementing Identity-Based Non-Discretionary Access Control

When formulating ideas for how to implement ORGCON, the discussion of the suitability of the O/G/W bit mask or similar mechanism arose repeatedly. Initially, there was reluctance to use these mechanisms because of the well-known weaknesses of traditional DAC mechanisms. The DAC access control list (ACL) and O/G/W mechanisms are useful, well known, and widely implemented. There is no apparent reason not to use them in implementing other policies. Confusion has sometimes resulted from the common identification of DAC mechanisms with DAC policy. The following discussion should aid in the clarification of the issue.

The particular category of controls we are interested in is the class of identity-based non-discretionary access controls, as exemplified by ORGCON. Traditional DAC mechanisms provide a weak form of need-to-know; the ORGCON policy requires a much stronger form of need-to-know. After considerable debate, we decided that the O/G/W mechanism can be an effective mechanism for identity-based access control, IF we also implemented supporting policy(ies) that closed the DAC weaknesses. Put another way, we designed the mechanisms to implement the ORGCON policy using the traditional DAC mechanisms in conjunction with other mechanisms. The uncontrolled copy and the lack of accountability weaknesses of DAC are limited by restricting user access to ORGCON objects to a limited set of functions.

A major difference between the requirements of the ORGCON policy and DAC mechanisms is delegation of authority. Under traditional DAC, authority to determine read and execute access to information is effectively given to anyone having read access to an object containing the information. Write access is somewhat more restricted. Under the ORGCON policy, the authority to grant read access is shared by two roles to whom authority is delegated. One role (the ORGREP) has the authority to change ACI associated with the object (i.e., the ACL). The other role (the RECREP) has the authority to change ACI which is part of the context (i.e., subject's group/role membership). This can be compared to mandatory controls wherein a single role (e.g., ISSO) or some agent such as the classification/clearance officer, changes ACI associated with the subject (clearance) and ACI associated with the object (classification).

The prototype implements two roles: the ORGREP and the RECREP. The prototype includes a "role" command, that allows users to assume a given role and limits their actions within that role. For example, a user wishing to act as the ORGREP would explicitly change role to that of ORGREP. Appropriate TCB checks are made to ensure that the user is authorized to act in the ORGREP role, and if so, the user, acting as ORGREP, is put in a restricted shell that limits the available commands to those necessary to perform the appropriate ORGREP functions (e.g., read ORGCON objects, add the distribution list, store the object, print the object). A similar role command is provided for the RECREP.

Since System V/MLS does not imlement ACLs it was necessary to develop a strategy for providing an ACL. Initially we anticipated using available space in the label structure to implement an ACL by creating a pointer field to a linked list containing the list of recipient organization roles. However, this proved infeasable due to the implementation of labels in System V/MLS. A further issue was how to notify recipients of a new ORGCON object and how to deal with recipients on remote AISs. Both of these were solved by exploiting the multi-level secure mail facility provided by System V/MLS. The ACL was incorporated in the header of the message and mail mechanisms are used to distribute ORGCON objects and notify recipients.

The credibility, reliability, and trustworthiness of the DAC authority is rather low. Lack of accountability undoubtedly contributes to this low esteem. While DAC is supposed to be used to implement need-to-know policy, the DAC Guideline [6] points out that access could be granted based on "whom do I like." Under the ORGCON policy and MAC, access is controlled by a designated authority who is held accountable for his/her action. This authority is responsible for changing the appropriate ACI based on information, such as a person's clearance or an organization's roster, supplied by equally authorized and audited officials.

5.4.2 Additional ORGCON ACI

To support the ORGCON policy, several attributes were added to the object's ACI. The ORGCON marking was previously discussed. The attribute "ORGCON- distribution" is also part of an object's ACI. The distribution list is a set of recipient organizations (e.g., CINCPAC, Division X), serving as an access control list within the computer system(s) and a distribution list when hardcopy is obtained. The definition of these organizations and roles is part of the ACC (i.e., the context on the destination AIS). The attribute originator identification (orig-ID) is also maintained in the ACI. Only the ORGREP can populate the ORGCON distribution list and provide the orig-ID for an ORGCON object. In the prototype, all the ORGCON-related ACI are associated with the object or are ACC (i.e., there is no additional ACI associated with the subject).

5.4.3 Designating an Object ORGCON-C

One of the components of the implementation of the ORGCON policy is the *ORGCON DESIGNATE program.* This program achieves the first steps in limiting the dissemination and use of ORGCON information.

Any user can create a potential ORGCON object. However, once an object is designated ORGCON, the creator of the object no longer has the authority over that object. The originating representative is responsible for attaching the distribution list to the object (although the creator may provide a suggested distribution list) and distributing the object. The designate program handles this "passing" of authority from the creator to the originating representative.

The program provides a convenient interface to the user for "passing" an object to an ORGREP, and handles the details associated with the changes in authority, labeling the object, and the restricted access requirements of an ORGCON-C object. The designate program takes the specified file and performs the following actions:

- Marks the file "ORGCON-C" (ORGCON candidate)
- Changes the user (creator) permissions to <read>.
- Changes the groups permissions to <read>.
- Changes the other permissions to <null>.
- Changes the group to the ORGREPS group.
- Renames the file uniquely to prevent accidental overwrite.
- Moves the file to /usr/users/orgreps.

In the process, the creator retains read permission on the file so (s)he may still review it, and the originating representatives may read the file (as long as the user is operating at the same MAC classification level as that of the object). By changing the group and moving the file to /usr/users/orgreps, access is restricted to the owner and the orgreps; no one in the group that the creator belongs to still has access to the file. This begins the process of changing the markings and putting the additional controls on the object. At this point, the owner still has limited authority over the ORGCON-C object.

5.4.4 Designating an Object ORGCON

The next step, then, is for the ORGREP to change the designation of an object from ORGCON-C to ORGCON. Once an object is designated ORGCON, the creator of the object no longer has the authority over that object. The program again provides a convenient interface to the object, and handles the details associated with the changes in ownership, labeling, and the restricted access requirements of an ORGCON object. At this point the designate program takes the specified file and performs the following actions:

- Marks the file "ORGCON"
- Changes the ownership to "ORGREP".
- Changes the owner permissions to <read>.
- Changes the groups permissions to <read>.
- Changes the other permissions to <null>.

The ORGREP then initiates the DISTRIBUTE program. This program prompts the ORGREP for a distribution list and handles the actual distribution of the ORGCON object. It checks to verify that the organizations specified as recipients are valid organizations and handles the dissemination of the object to local and remote systems as indicated in configuration files. (Maintaining this authorized list of valid organizations is outside the scope of the prototype). This program likewise provides a convenient interface for the ORGREP to distribute an ORGCON object.

6 CONCLUSIONS

6.1 Concerning Security Policies

In this paper, a policy named ORGCON (Organization Controlled) that is based on the ORCON policy has been defined. Though a policy for manual control of paper documents can be workable even though vague or lacking detail, the policy must be extended and the detail must be specified to make it suitable for an AIS. By creating supporting policies and hypothesizing procedures, the ORCON policy was extended and details added to create the ORGCON policy. The ORGCON policy created permits copying for distribution but not for incorporation of information in derivative objects.

Since the prototype described in this paper was a proof-of-concept, conformance to real world constraints was not the highest priority though we understand that a real implementation would indeed have many such constraints. Experience suggests that most organizations do not understand their information flows, and that security restrictions exacerbate the concerns. In the extreme, some organizations may decide not to automate certain security policies because the AIS will not have the ability to discern when the letter of the law may be ignored with impunity. However, we believe that it is both possible and desirable to implement additional security policies in an AIS and plan to implement other existing information dissemination/control policies.

6.2 Concerning Technology

This effort has demonstrated that it is possible to implement additional security policies by extending an existing TCB. Such an effort requires a well formulated approach such as the Generalized Framework for Access Control. Part of our approach has involved formal modeling, which proved invaluable in aiding our understanding of the policies. A new understanding of the increased level of detail required for modeling GFAC concepts will be reported in a subsequent paper.

As with all research, additional questions surfaced while several others were answered. In particular, the potential growth in size and complexity of the TCB if the mechanisms for implementing all of the security policies are placed in the same TCB remains an issue. Exploration of the relationships among TCB mechanisms supporting separate policies is required. For example, the TCB code that implements the ORGCON controls has no relationship to MAC or DAC policy. Part of the problem is determining appropriate terminology to express the concepts. What words should be used to refer to mechanisms that implement different policies? Can and should the TCB concept be expanded to embrace additional policies? What should be the relationships among TCBs for different policies? It is anticipated that answers to at least some of these questions will be discovered in the coming year.

LIST OF REFERENCES

1. Abrams, M. D., K. W. Eggers, L. J. LaPadula, I. M. Olson, "A Generalized Framework for Access Control: An Informal Description," *Proceedings of the 13th National Computer Security Conference*, October 1990.
2. Director of Central Intelligence Directive No. 1/7, *Control of Dissemination of Intelligence Information*, 4 May 1981.
3. Graubart, T. D., "On the Need for a Third Form of Access Control," *Proceedings of the 12th National Computer Security Conference*, Baltimore, MD, October 1989.
4. LaPadula, L. J., "Formal Modeling in a Generalized Framework for Access Control," *Proceedings of the Computer Security Foundation Workshop III*, 12 June 1990.
5. National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985.
6. National Computer Security Center, *A Guide to Understanding Discretionary Access Control in Trusted Systems*, NCSC-TG-003, Version-1, 30 September 1987.
7. National Computer Security Center, *Final Evaluation Report of American Telephone and Telegraph System V/MLS Release 1.1.2 Running on UNIX System V Release 3.1.1*, CSC-EPL-89/003, 18 October 1989.
8. Williams, J. C. and M. L. Day, "Sensitivity Labels and Security Profiles," *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988.
9. AT&T, "System V/MLS Users' Guide and Reference Manual," 27 March 1990.

HONEST DATABASES THAT CAN KEEP SECRETS

Ravi S. Sandhu and Sushil Jajodia[*]

Center for Secure Information Systems
and
Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030-4444

ABSTRACT Polyinstantiation has generated a great deal of controversy lately. Some have argued that polyinstantiation and integrity are fundamentally incompatible, and have proposed alternatives to polyinstantiation. Others have argued about the correct definition of polyinstantiation and its operational semantics. In this paper we provide a fresh analysis of the basic problem that we are trying to solve, i.e., how can a honest database keep secrets? Our analysis leads us to the concept of restricted polyinstantiation wherein we show how to solve this problem without compromising on any of the following requirements: secrecy, integrity, availability-of-service, element-level labeling and high assurance. This is the first solution to meet all these requirements simultaneously.

1 INTRODUCTION

What distinguishes a multilevel database from ordinary single level ones? In a multilevel world as we raise a user's clearance new facts emerge; conversely as we lower a user's clearance some facts get hidden. Therefore users with different clearances see different versions of reality. Moreover, these different versions must be kept coherent and consistent—both individually and relative to each other—without introducing any downward signaling channels.[†]

The caveat of "no downward signaling channels" poses a major new problem in building multilevel secure database management systems (DBMSs) as compared to ordinary single-level DBMSs. This caveat is inescapable and absolute. We must reject outright "solutions" which tolerate downward signaling channels. Solutions with such channels, e.g., as proposed in [1, 9], may well be acceptable as an engineering compromise in particular situations. But they are clearly not acceptable as general-purpose solutions. This point needs to be emphasized because security is usually the one to take the first hit in engineering trade-offs. It behooves us as security researchers to present solutions which avoid taking this hit while at the same time providing

- no downward signaling channels,
- consistency and integrity of the database both within and across levels,
- flexibility for application semantics,
- fine-grained classification of data (i.e. element-level labeling), and
- high assurance with minimal trusted code.

[*]The work of both authors was partially supported by the U.S. Air Force, Rome Air Development Center through subcontract #C/UB-49;D.O.No.0042 of prime contract #F-30602-88-D-0026, Task B-O-3610 with CALSPAN-UB Research Center.

[†]We deliberately use the term downward signaling channel rather than covert channel. A downward signaling channel is a means of downward information flow which is inherent in the data model and will therefore occur in *every* implementation of the model. A covert channel on the other hand is a property of a specific implementation and not a property of the data model. In other words, even if the data model is free of downward signaling channels, a specific implementation may well contain covert channels due to implementation quirks.

The central point of this paper is to demonstrate how these diverse goals can be met in a multilevel relational DBMS without compromising security as part of the bargain. Our solution is simple in concept and almost obvious in retrospect. For the most part it uses standard concepts from the database arena. A key new idea is to introduce a special value called "restricted" distinct from the normal data values of an attribute (or column) as well as distinct from "null." The value "restricted" denotes that the particular field cannot be updated at the specified level. So long as the value of a field is not "restricted" our multilevel relations behave much as ordinary single-level relations do. Particular attention is required when a field is changed from unrestricted to restricted and vice versa. A notable property of our solution is that it can be implemented entirely by untrusted subjects, i.e., subjects which are not exempted from the simple security or $\star$-properties.[‡]

The rest of this paper is organised as follows. Section 2 reviews the concept of polyinstantiation from an intuitive point of view, with the objective of identifying the sources of polyinstantiation and alternatives to it. Section 3 informally introduces our solution of restricted polyinstantiation and illustrates it by examples. Section 4 formalises and precisely defines our solution. It also provides additional examples. Section 5 discusses how our solution can provide the highest degree of assurance. Section 6 concludes the paper.

2 POLYINSTANTIATION

The concept of polyinstantiation was explicitly introduced by Denning et al [3] in connection with the SeaView project. Since then much has been written about this topic [1, 3, 4, 5, 6, 7, 9, for instance]. In this paper we will set aside all this previous theory, formalism and debate. Instead we go back to first principles and consider by means of examples how polyinstantiation arises and therefore how it might be controlled. We assume the reader is familiar with basic relational notions and terminology.

2.1 The Source of Polyinstantiation

Polyinstantiation can occur in basically two different ways which we call *polyhigh* and *polylow* respectively for mnemonic convenience.

1. Polyhigh occurs when a high user[§] attempts to insert data in a field which already contains low data. Overwriting the low data in place will result in a downward signaling channel. Therefore the high data can be inserted only by creating a new instance of the field to store the high data. We also have the option of rejecting the update altogether with the attendant possibility of denial-of-service to the high user.

2. Polylow occurs in the opposite situation where a low user attempts to insert data in a field which already contains high data. In this case rejecting the update is not a viable option because it establishes a downward signaling channel. That leaves us two alternatives. We can overwrite the high data in place which violates the integrity of the high data. Or we can create a new instance of the field to store the low data.

In both cases note that we have identified "secure" alternatives to polyinstantiation. These alternatives are secure in the sense of secrecy and information flow. Unfortunately the alternatives have denial-of-service and integrity problems reiterated below.

[‡]The protocols of section 4 can be simplified if trusted subjects which are exempted from these properties are allowed in selected situations.

[§]Strictly speaking we should be saying subject rather than user. For the most part we will loosely use these terms interchangeably. Where the distinction is important we will be appropriately precise.

1. The alternative to polyhigh entails denial-of-service to high users by low users (i.e., once a low value has been entered in a field a high value cannot be entered until the low value has been nullified by a low subject[¶]).
2. The alternative to polylow entails destruction of high data by low users which presents a serious integrity problem (i.e., the high data is overwritten in place by low data.)

A naive implementation of these alternatives will create more real security problems than it solves. Our main contribution in this paper is to show how these alternatives to polyhigh and polylow can be employed in a careful, disciplined manner to achieve secrecy, availability-of-service and integrity with high assurance.

It should be noted that there is an important difference between polyhigh and polylow. Polyhigh can be completely prevented by reactive mechanisms at the cost of denial-of-service to entry of high data. This is likely to be a tolerable cost in many applications. On the other hand polylow cannot be completely prevented by reactive mechanisms. At the moment of enforcement a reactive mechanism has only the alternative of overwriting high data by low data. This is likely to be intolerable in most applications. Therefore polylow must—for all practical purposes—be prevented by a proactive mechanism, i.e., steps must be taken in advance of the problem's occurrence to ensure that it cannot occur.

2.2 Polyhigh Example

Let us now consider a concrete example to make polyhigh and polylow clearer. Consider the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Here, as in all our examples, each attribute in a tuple not only has a value but also a classification. In addition there is a tuple-class or TC attribute. This attribute is computed to be the least upper bound of the classifications of the individual data elements in the tuple.

Now consider the following scenario.

1. A U user updates the destination of the Enterprise to be Talos. The relation is therefore modified as follows.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

2. Next a S user attempts to modify the destination of the Enterprise to be Rigel. We cannot overwrite the destination in place because that would create a downward signaling channel. We can reject the update at the risk of denying entry of legitimate secret data. Or we can polyinstantiate and modify the relation to appear as follows, respectively for U and S users. Note that U users see no change.

[¶]This protocol—of nullifying low data prior to entry of high data—does not guarantee protection against denial-of-service. If a low value is nullified to enable entry of a high value there remains the risk that a low Trojan Horse can enter another low data value before the high subject has the opportunity to enter its high value. The solution described in this paper (see Section 3) eliminates this vulnerability.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Exploration	U	Rigel	S	S

What are we to make of this last relation given above. There are at least two reasonable interpretations.

- *Cover Story.* The destination of Talos may be a cover story for the real destination of Rigel. In this case the database is accurately mimicking the duplicity of the real world. There are, however, other ways of incorporating cover stories besides polyinstantiation. For example we may have two attributes, one for cover-story destination and one for the real destination. Debate on the relative merits and demerits of these techniques is outside the scope of this paper. *For purpose of this paper we assume that polyinstantiation is not to be used for cover stories. We therefore reject this alternative as a valid interpretation.*

- *Temporary Inconsistency.* We have a temporary inconsistency in the database which needs to be resolved. For instance the inconsistency may be resolved as follows: the S user who inserted the Rigel destination latter logs in at the U level and nullifies the Talos value, so thereafter the relation appears respectively as follows to U and S users.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

It is most important to understand that this scheme does not create a downward signaling channel from one subject to another. The nullification of the destination at the U level is being done by a U subject. One might argue that there is a downward signaling channel with a human in the loop. The human is however trusted not to let the channel be exercised without good cause. Finally note that the U user who executed step 1 of the scenario may again try to enter Talos as the destination, which brings us within the scope of polylow.

2.3 Polylow Example

Our example for polylow is similar to the polyhigh example with the difference that the two update operations occur in the opposite order. So again consider the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

This time consider the following scenario.

1. A S user modifies the destination of the Enterprise to be Rigel. The relation is modified to appear respectively as follows to U and S users. Note that U users see no change in the relation.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

2. A U user updates the destination of the Enterprise to be Talos. We cannot reject this update on the grounds that a secret destination for the Enterprise already exists, because that amounts to establishing a downward signaling channel. We can overwrite the destination field in place at the cost of destroying secret data. This would give us the following relation for both U and S users.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

For obvious reasons this alternative has not been seriously considered by most researchers. That leaves us the option of polyinstantiation which will modify the relation at the end of step 1 to the following for U and S users respectively.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Exploration	U	Rigel	S	S

This is exactly the same relation as obtained at the end of step 2 in our polyhigh example. The possible interpretations are therefore similar, i.e., we either have a temporary inconsistency or a cover story (the latter alternative has already been rejected for our database). The temporary inconsistency can be corrected by having a U subject (possibly created by a S user logged in at the U level) nullify the Talos destination. But the inconsistency may recur again and again.

3 RESTRICTED POLYINSTANTIATION

In the previous section we have examined the source of polyinstantiation and identified polyhigh and polylow as the two different ways in which polyinstantiation arises. In this section we consider applications which have the following requirements.

1. Downward signaling channels cannot be tolerated.
2. The simple security and $\star$-properties must be enforced for all subjects, i.e., no trusted code can be used.
3. Temporary inconsistencies cannot be tolerated.
4. Denial of data entry service to high users cannot be tolerated.

Moreover each of these requirements has equal importance and one cannot be sacrificed for another. The scenarios of the polyhigh and polylow examples of the previous section show that polyinstantiation by itself cannot meet these requirements simultaneously. One requirement or the other must give in some way.

In this section we show how all four requirements identified above can be simultaneously met. We describe our solution as *restricted polyinstantiation*. The basic idea is to introduce a special symbol denoted by "restricted" as the possible value of a data element. The value "restricted" is distinct from any other value for that element and is also different from "null." In other words the domain of a data element is its natural domain extended with "restricted" and "null." We define the semantics of "restricted" in such a way that we are able to eliminate both polyhigh and polylow. "Null" has exactly the same semantics as any other data value and needs no special treatment.

Let us now play out the polyhigh and polylow scenarios of the previous section to intuitively motivate our solution. A formal description of the update protocols is given in the next section.

3.1 Polyhigh Example Revisited

Consider again the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Now consider the following scenario.

1. A U user updates the destination of the Enterprise to be Talos. The relation is therefore modified as follows.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

2. Next a S user attempts to modify the destination of the Enterprise to be Rigel. We cannot polyinstantiate even temporarily, so we must reject this update. Do we have denial-of-service to the S user? No, because the S user can obtain service as follows.

   Step 2a. The S user first logs in as a U-subject and marks the destination of the Enterprise as restricted giving us the following relation.[||]

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

   The meaning of restricted is that this field can no longer be updated by a U user. U users can therefore infer that the true value of Enterprise's destination is classified at some level not dominated by U.

   Step 2b. The S user then logs in as a S-subject and enters the destination of the Enterprise as Rigel giving us the following relations at the U and S levels respectively.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

[||]Alternately the S user logs in at the U-level and requests some properly authorized U user to carry out this step. Communication of this request from the S user to the U user may also occur outside of the computer system, by say direct personal communication or a secure telephone call.

How does this differ from the scenario of section 2.2 (where the end result after cleaning up the temporary inconsistency was as above except that we have null instead of restricted)? The main difference is that, after step 2a, U users are no longer able to update the destination of the Enterprise. In particular, attempts by U users to reenter Talos as the destination of Enterprise will be rejected on the grounds that the field is restricted. Therefore the relation is guaranteed to be consistent till such time as the restricted value is eliminated. Consideration of who should be allowed to enter and remove the restricted value is deferred for now.

Does step 2a introduce a signaling channel? Yes, but this signaling channel is very similar to the one resulting from the nullification of Talos at the U-level in the example of section 2.2. Both involve a trusted S user in the loop who presumably will ensure that the channel is not exercised wantonly, but rather that this inference is permitted only when the real world situation is actually so. Such a channel with trusted humans in the loop can be exercised only by Trojan Horses that are capable of manipulating the real world. This entails the manipulation of real trusted people making real decisions and not merely the manipulation of bits in a database.

3.2 Polylow Example Revisited

Now consider the two update operations in the opposite order. So again we begin with the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

This time consider the following scenario.

1. A S user modifies the destination of the Enterprise to be Rigel. This update is rejected! Instead the S user is asked to go through steps 2a and 2b of section 3.1 giving us the following relations at the U and S levels respectively.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

2. A U user updates the destination of the Enterprise to be Talos. The update is rejected on the grounds that the field is restricted.

Note that there is no denial-of-service to the S user. What is happening is a denial of improper service, i.e., there is a protocol for entering high data which all S users are required to follow. Failure to follow the protocol results in denial-of-service but this can hardly be considered a security breach. The denial-of-service to the U user is, of course, only appropriate in this situation.

There is a crucial difference between this protocol and the one discussed in section 2.1. In both cases entry of high data is enabled by an action of a low subject. Our protocol requires the low subject to enter the "restricted" value in the data element. In section 2.1 the suggestion was for the low subject to enter a "null" value. The key difference in the two cases is that a null value can be made non-null by a low Trojan Horse, whereas the restricted value cannot be made unrestricted by a low Trojan Horse. The latter operation requires a special privilege whose distribution is carefully controlled by non-discretionary means. This privilege is available only to selected low subjects who are trusted to exercise its use properly.

4 THE PREVENT PROTOCOLS

In this section we precisely define the collection of update protocols illustrated by example in the previous section. We collectively call this collection the *prevent protocols* because they prevent polyinstantiation due to either polyhigh or polylow from occurring. These protocols can be implemented entirely by untrusted subjects, i.e., subjects which are not exempted from the simple security or $\star$-properties.

4.1 Multilevel Relations

We begin by reviewing some basic concepts and notation for multilevel relations. Let A_1, C_1, A_2, C_2, ..., A_n, C_n denote the attributes (columns) of a multilevel relation R with element level labeling. Each A_i is a *data attribute* and each C_i is the *classification attribute* for A_i. A data attribute can take on values from its natural domain D_i extended with two special values, "null" and "restricted," whose meaning will be defined shortly. We assume that each C_i can take on any value c in the security lattice.** We require that C_i cannot be null. Finally R has a collection of *relation instances* R_c one for each access class c in the given lattice.

Assume there is a user-specified primary key AK consisting of a subset of the data attributes A_i. We call AK the *apparent primary key* of the multilevel relation scheme. In general AK will consist of multiple attributes. We have the following requirement in analogy to entity integrity in the standard relation model. (The notation $t[A_i]$ denotes the value of the A_i attribute in tuple t, and similarly for $t[C_i]$.)

Property 1 [Entity Integrity] Instance R_c of R satisfies entity integrity iff for all $t \in R_c$: (i) AK is uniformly classified in each tuple, i.e., $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j]$, and (ii) the classification of each non-key data attribute dominates the classification of the apparent key, i.e., $A_i \notin AK \Rightarrow t[C_i] \geq t[C_{AK}]$ where C_{AK} is the classification of AK. □

The notions introduced thus far are standard ones first introduced in the SeaView model [7]. Our next requirement severely limits polyinstantiation and distinguishes the approach of this paper from previous work on element-level labeling (such as [3, 4, 5, 6, 7]).

Property 2 [Key Integrity] R satisfies key integrity iff for every R_c we have for all $i: AK, C_{AK} \rightarrow A_i, C_i$. □

This property stipulates that the user-specified apparent key AK, in conjunction with key-classification C_{AK}, functionally determines all other attributes. In other words R_c cannot have more than one tuple for a given combination of values for AK and C_{AK}. That is, the real primary key of the relation is AK, C_{AK}. The effect of key integrity is to rule out instances such as the following.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Exploration	U	Rigel	S	S

The reason for rejecting this instance is its inconsistency in specifying two different destinations—one secret and one unclassified—for the Enterprise. Recall our assumption that cover stories are not to be incorporated by polyinstantiation, so interpretations such as discussed in [5] do not apply in this situation. Key integrity does allow instances such as the following where there is polyinstantiation of the key.

**In practice of course it is desirable to place appropriate upper and lower bounds on each C_i. This will only require minor changes to the following discussion.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	S	Spying	S	Rigel	S	S

In this case we interpret the two tuples as describing two distinct Starships which happen to have the same name.

The next property is concerned with consistency between relation instances at different access classes. Here again we depart from the analogous property defined in [5, 6, 7].††

Property 3 [**Inter-Instance Integrity**] R satisfies inter-instance integrity iff for all $c' \leq c$ we have $R_{c'} = \sigma(R_c, c')$ where the *filter function* σ produces the c'-instance $R_{c'}$ from R_c as follows:

1. For every tuple $t \in R_c$ such that $t[C_{AK}] \leq c'$ there is a tuple $t' \in R_{c'}$ with $t'[AK, C_{AK}] = t[AK, C_{AK}]$ and for $A_i \notin AK$

$$t'[A_i, C_i] = \begin{cases} t[A_i, C_i] & \text{if } t[C_i] \leq c' \\ <\text{restricted}, c'> & \text{otherwise} \end{cases}$$

2. There are no tuples in $R_{c'}$ other than those derived by the above rule. □

The filter function maps a multilevel relation to different instances, one for each descending access class in the security lattice. Filtering limits each user to that portion of the multilevel relation for which he or she is cleared. For instance filtering the following S-instance of SOD

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

gives us the following U-instance

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

4.2 Update Protocols

In section 4.1 we have identified integrity properties for multilevel relations considered at some instant in time as static objects. We now consider the dynamic behavior of these relations by considering their update semantics. We emphasize that our protocols do not require any exception from the simple security or $\star$-properties.‡‡ There are three subcases to consider as follows.

4.2.1 Data Value Update

By the term *data value* we mean any value other than "restricted." Our first protocol addresses the case where the value of attribute $t[A_i]$ is changed from its previous data value to a new data value, i.e., neither the previous value nor the new one can be "restricted." "Null" does not need any special treatment in our protocols and is viewed as just another data value. We have the following update protocol.

††The definition of the filter function given in [5, 6, 7] differs from the one given here in that <restricted,c'> is replaced by <null,$t[C_{AK}]$>.

‡‡Note that the protocols can be simplified if trusted subjects which are exempted from these properties are allowed in selected situations. In particular the protocol to change a restricted value to unrestricted (see section 4.2.3) would be considerably simplified by using a trusted subject which is exempted from the $\star$-property.

Protocol 1 $t[A_i]$ can be changed from its previous data value to a new data value by a c-user only if $t[C_i] = c$.

The effect of this update operation is defined as follows.

1. The value of $t[A_i]$ is changed to its new value in all relation instances $R_{c'}$, $c' \geq c$. The value of $t[C_i]$ remains unchanged as c in all $R_{c'}$, $c' \geq c$.
2. All other instances of R remain unchanged. □

Note that the precondition for this protocol is stated as a necessary condition ("only if"). It is thus a mandatory requirement. In addition to this mandatory pre-condition we may as usual impose further mandatory and/or discretionary controls.

To illustrate the protocol consider the following U and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

An update by a U user to change the Objective from "Exploration" to "Mining" has the following effect.

Starship	Objective	Destination	TC
Enterprise U	Mining U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Mining U	Rigel S	S

That is the update takes effect at both the U and S levels. An attempt by a S user to change the Objective attribute would be rejected. So would an attempt by a U user to change the Destination attribute. A S user may change the Destination attribute to say "Talos" giving us the following U and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Mining U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Mining U	Talos S	S

To appreciate how "null" is treated just like any other data value consider what happens if a S user nullifies the Destination attribute. We get the following U and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Mining U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Mining U	null S	S

The Destination attribute remains restricted for U users and the null value is shown only to S users. The classification of the null at S signifies that data in this field can only be entered by S users. If the Destination attribute has a null value at the U level then both U and S instances of SOD must be as follows.

Starship		Objective		Destination		TC
Enterprise	U	Mining	U	null	U	U

In this case U users are allowed to enter data for the Destination attribute whereas S users are not permitted to do so. In order to enable S users to change the Destination of the Enterprise we must first restrict this field at the U level. This brings us to our next protocol.

4.2.2 Update from Unrestricted to Restricted

Let us first consider the case where the security lattice is totally ordered (i.e., there are no compartments). An update of attribute A_i in tuple t from some existing data value to "restricted" is performed as follows.

Protocol 2 $t[A_i]$ can be changed from its previous data value to "restricted" by a c-user only if $t[C_i] = c$.
The effect of this update operation is defined as follows.

1. The value of $t[A_i, C_i]$ is changed to $<$restricted, $c>$ in the instance R_c.
2. Let $\pi(c)$ be the immediate predecessor of c (i.e., $\pi(c) > c$ and there is no c' such that $\pi(c) > c' > c$). The value of $t[A_i, C_i]$ is changed to $<$null, $\pi(c)>$ in all instances $R_{c'}, c' > c$.
3. All other instances of R remain unchanged. □

It suffices to have the pre-condition $t[C_i] = c$ for this operation because, in conjunction with the inter-instance integrity property, $t[C_i] = c$ implies

$$(\forall c' : t[C_{AK}] \leq c' < c)\ t[A_i, C_i] = <\text{restricted}, c'> \text{ in } R_{c'}$$

In other words a data element can be made restricted at level c only if its data value is currently classified at level c, which in turn implies that the data element is restricted at all relevant levels below c.

To illustrate the effect of such updates consider the following U instance of SOD (which is identical to the S instance).

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	U	U

A U user can change the destination of the Enterprise to be "restricted" giving us the following U and S instances.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	S	S

Now let us consider the general case of a partially ordered security lattice. The problem with partially ordered labels lies in step 2 in defining the effect of protocol 2. In a partial ordering there may be multiple immediate predecessors of c so $\pi(c)$ is no longer uniquely defined. As part of the update operation we have to designate one of c's immediate predecessors as the distinguished one which will remain unrestricted. All other immediate predecessors become restricted. Let $\pi(c)$ denote the distinguished immediate predecessor. Step 2 of protocol 2 needs to be restated as follows.

2$'$. The value of $t[A_i, C_i]$ is changed as follows for all instances $R_{c'}, c' > c$.

$$t[A_i, C_i] = \begin{cases} <\text{null}, \pi(c)> & \text{if } c' \geq \pi(c) \\ <\text{restricted}, c'> & \text{if } c' \not\geq \pi(c) \end{cases}$$

As an example consider a lattice with four labels, S, U, M_1 and M_2; where M_1 and M_2 are both dominated by S and both dominate U, but M_1 and M_2 are themselves incomparable. Suppose we have the following instance of SOD at all four levels.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	U	U

Let a U user make the Destination field of the Enterprise "restricted" while designating M_1 to be $\pi(U)$ for this update. The U, M_1, M_2 and S instances of SOD will respectively become as follows.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	M_1	M_1

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	M_2	M_2

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	M_1	M_1

4.2.3 Update from Restricted to Unrestricted

Again for simplicity let us first consider the case where the lattice is totally ordered. We have the following protocol for making a field unrestricted.

Protocol 3 $t[A_i]$ can be changed from its current value of "restricted" to a data value dv only by a c-user.

The effect of this update operation is defined as follows.

1. The value of $t[A_i, C_i]$ is changed to $<dv, c>$ in all instances $R_{c'}$, $c' \geq c$.

2. All other instances of R remain unchanged. □

The pre-condition for this update, that $t[A_i, C_i] = <\text{restricted}, c>$ in R_c, is sufficient to ensure that $t[A_i, C_i] = <\text{restricted}, c'>$ in all R'_c, $c' \leq c$ (due to inter-instance integrity).

The protocol will overwrite any existing data value for $t[A_i]$ in instances R'_c, $c' > c$. This operation therefore has the potential for creating integrity problems by overwriting existing higher level data. We have rejected this approach as a general solution in section 2. Here we are proposing to employ it for the specific purpose of converting a field from restricted to unrestricted. We require that this be a specially privileged operation so that we can be sure it is executed only when the real world conditions warrant it. We will return to this point in the next section.

To illustrate this operation consider the following U and S instances of SOD.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	S	S

A suitably privileged U user can change the value of the Destination attribute in this tuple to be say "Talos" giving us the following (identical) U and S instances of SOD.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

Next let us consider the case of a partially ordered security lattice. The pre-condition of protocol 3 is no longer sufficient. Before a c user is allowed to change a restricted field to non-restricted we must ensure that field is restricted at all levels which do not dominate c. This includes levels which are dominated by c as well as levels incomparable with c. The latter requirement cannot be checked by a c user without violating simple-security. We circumvent this problem by requiring the update of protocol 3 to occur in two phases as follows.

1. *Preparatory Phase.* Login at level $t[C_{AK}]$ and set

$$t[A_i, C_i] = <\text{restricted}, c'> \quad \text{in all instances } R'_c,\ c' \geq t[C_{AK}]$$

   i.e., set $t[A_i]$ to "restricted" at all levels where tuple t is visible.

2. *Update Phase.* Login at level c and set $t[A_i, C_i] = <dv, c>$.

The net effect of this modified protocol is to set

$$t[A_i, C_i] = \begin{cases} <dv, c> & \text{in all instances } R_{c'},\ c' \geq c \\ <\text{restricted}, c'> & \text{in all instances } R_{c'},\ c' \not\geq c \end{cases}$$

For example consider the following U, M_1, M_2 and S instances of SOD respectively taken from the end of section 4.2.2.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	M_1	M_1

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	M_2	M_2

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	M_1	M_1

The preparatory phase will give us the following U, M_1, M_2 and S instances of SOD respectively.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	M_1	M_1

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	M_2	M_2

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	M_2	M_2

In other words the preparatory phase restricts the Destination attribute of this tuple at all levels above U (which is the key class of the tuple). Subsequently, the update phase results in (say) the following U, M_1, M_2 and S instances of SOD respectively.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	M_1	M_1

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	M_2	M_2

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	M_2	M_2

5 ASSURANCE

In this section we briefly consider how the prevent protocols can be enforced.

Our first observation is that all our protocols adhere to both simple security and the $\star$-property. They can therefore be enforced by a DBMS trusted computing base (TCB) to the highest assurance standards without the use of subjects which are exempt from simple-security or the $\star$-property.

Secondly, our protocols are designed to achieve integrity and availability-of-service in addition to secrecy. The secrecy objective can be enforced to A1 standards by strict enforcement of simple security and the $\star$-properties. In order to achieve the integrity and availability of service requirements we need controls beyond the traditional simple security and $\star$-property. Let us consider each of the following three cases in turn.

5.1 Data Value Update

This is the simplest case where our multilevel relations behave much as conventional single-level relations do. It is obvious that in a high integrity system updates must be carefully controlled even within a single security level. Conventional databases use mechanisms such as well-formed transactions and least privilege for this purpose [2, 8]. The DBMS TCB must provide high assurance support for such mechanisms. We do not need any additional mechanisms for multilevel DBMSs. The required mechanisms should anyway be available in high-quality single-level DBMSs as discussed in [8].

5.2 Update from Unrestricted to Restricted

Assigning a restricted value to a field with classification *c* requires a check that this field is already restricted at levels below *c*. This is feasible within the scope of simple security. In high assurance systems this application-independent pre-condition should be checked by the DBMS TCB. At lower levels of assurance the pre-condition may be tested by individual transactions rather than the DBMS.

The effect of restricting a field at the *c* level is dangerous in that it can cause denial-of-service to *c* users. So when the destinations of all our flights are made restricted, when they should not be, we might end up grounding the entire fleet! Therefore the ability to mark a field as restricted should be a carefully controlled privilege. This privilege should be assigned to a few subjects who need to do this operation. We can ensure that this privilege cannot be acquired except by some very special non-discretionary means such as involving intervention by a security officer.

The general problem of incorrect data essentially exists whether or not we recognize restricted as a special value. For suppose a malicious program running at the U level, and obeying simple security and $\star$-property, sets the destination of all flights to be Dayton, Ohio. Does the entire fleet converge on Wright Patterson Air Force Base? Presumably a high integrity system has corrective measures to detect and recover from such errors. In principle, incorrectly restricted fields present a similar problem except that recovery may be slightly more cumbersome.

5.3 Update from Restricted to Unrestricted

An update from restricted to unrestricted is different from the previous two cases because we cannot test the pre-conditions for this action within the confines of simple security. If we wish to prevent overwriting of high data by this operation we have to check that no high data exists (i.e., no non-null high data exists). In view of simple security this is not feasible. Therefore we define the operation as potentially overwriting high data. It follows that we must strictly control the ability to make a restricted value unrestricted. The control in this case should be even stricter than in the case of update from unrestricted to restricted. Alternately, we can use a trusted subject for this operation.

6 CONCLUSION

In this paper we have shown how both the polyhigh and polylow variations of polyinstantiation can be eliminated by our solution of restricted polyinstantiation. This allows us to avoid downward signaling channels, inconsistencies, denial of data entry to high users and the overwriting of high data by low subjects while providing element-level labeling. This is the first solution to meet all these requirements simultaneously.

In conclusion we wish to note that restricted polyinstantiation makes a particular trade-off among conflicting objectives. It may be eminently suitable to most applications. Yet we would advise against having this as the only option. Databases are long lived and develop a great deal of inertia over their life. Moreover different applications may call for different trade-offs. For example temporary inconsistencies may be preferred to inconvenience in data entry. General-purpose multilevel secure DBMSs must cater to such applications too. Therefore our recommendation is that restricted polyinstantiation be available as one of several options that a multilevel secure DBMS supports.

Acknowledgment

We are indebted to John Campbell, Joe Giordano, and Howard Stainer for their support and encouragement, making this work possible. The opinions expressed in this paper are of course our own and should not be taken to represent the views of these individuals.

References

[1] Burns, R.K. "Referential Secrecy." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 133-142.

[2] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy*, 184-194 (1987).

[3] Denning, D.E., Lunt, T.F., Schell, R.R., Heckman, M., and Shockley, W.R. "A Multilevel Relational Data Model." *IEEE Symposium on Security and Privacy*, 220-234 (1987).

[4] Denning D.E. "Lessons Learned from Modeling a Secure Multilevel Relational Database System." In *Database Security: Status and Prospects*, (Landwehr, C.E., editor), North-Holland, 35-43 (1988).

[5] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 104-115.

[6] Jajodia, S., Sandhu, R.S. and Sibley, E. "Update Semantics for Multilevel Relations." *Sixth Annual Computer Security Applications Conference*, Tucson, Arizona, December 1990, pages 103-112.

[7] Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M. and Shockley, W.R. "The SeaView Security Model." *IEEE Transactions on Software Engineering*, 16(6):593-607 (1990).

[8] Sandhu, R.S. and Jajodia, S. "Integrity Mechanisms in Database Management Systems." *13th NIST-NCSC National Computer Security Conference*, Washington, D.C., October 1990, 526-540.

[9] Wiseman, S.R. "On the Problem of Security in Data Bases." In *Database Security III: Status and Prospects*, (Spooner, D.L. and Landwehr, C.E., editors), North-Holland, pages 143-150 (1990). Also available as Royal Signal and Radar Establishment, U.K., Memo 4263.

IDENTIFYING AND CONTROLLING UNDESIRABLE PROGRAM BEHAVIORS

Maria M. King*
MKing@Dockmaster.ncsc.mil

Abstract

This paper describes a new mechanism for comparing selected program properties against a policy, or set of rules, that states allowable program behavior[2, 10]. The motivation for this work is the increased need to control undesirable behaviors of programs, such as those inherent in Trojan horses and computer viruses. This mechanism, called an Automatic Policy Checker (APC), is currently implemented under SunOS[1]*. This paper will discuss the design and implementation of the APC and the application of the APC to the virus problem. Conclusions concerning anti-viral policy in light of the test results will also be presented.*

Introduction

The motivation for this work is the increased need for computer security mechanisms to control undesirable activity of programs, such as those caused by computer viruses[1], Trojan horses and other types of malicious logic.

The major contribution of this work is an automatic tool, called an Automatic Policy Checker (APC), for comparing certain types of program behaviors against a policy that states allowable program behaviors. An important feature of the APC is that it does not implement any specific policy, clearly separating the policy from the mechanism which enforces the policy[8]. Existing mechanisms either rely on the user to specify their own policy[7] or embed an ad hoc policy in the mechanism[5]. The APC allows experiments with policies intended to prohibit a variety of undesirable program behaviors. The APC does not rely on any new architectural support, has minimal effect on performance, and does not require user knowledge of threat. Furthermore, if the APC is used in conjunction with a filter mechanism as described in [2, 6], reliance on some number of humans to act in a trustworthy manner, which is often required in many computer security mechanisms, is no longer needed.

This paper first describes a formal language based on regular expressions that was developed for stating policies and certain types of program behaviors. A high-level overview of the design of the APC is described here while [10] provides a more detailed discussion. The APC has been applied to the computer virus problem. A study of anti-viral policies based on the viral property of *file modification* was conducted and is described in the section on policies. Experiments were run and the empirical data is discussed and results presented.

High-Level Overview

The idea is to explicitly state a system's policy regarding allowable program activity. Subsequently, the APC is used to compare a selected program property against the policy, prior to installation. The APC determines whether a program's specified actions fall within the perimeter of a particular policy.

Definition 1 *A policy is a set of rules that formally states allowable program behavior, in a particular system.*

*Formerly Maria M. Pozzo.
[1]SunOS is a trademark of Sun Microsystems, Incorporated.

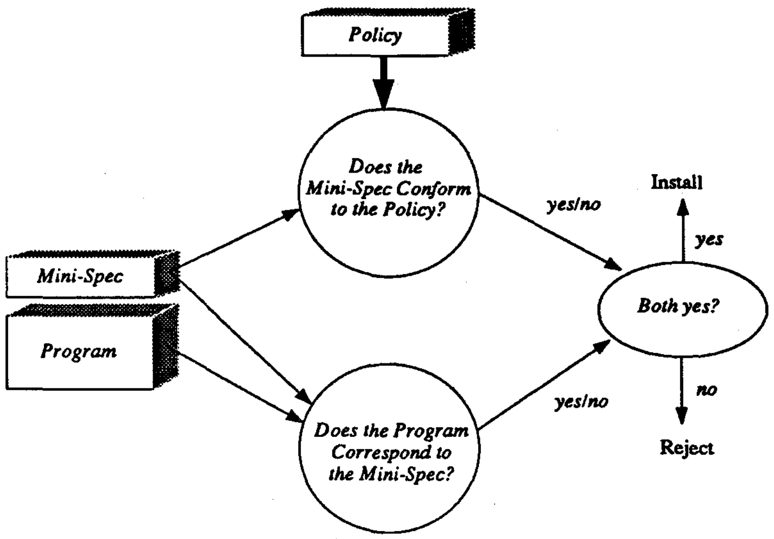

Policy
Does the Mini-Spec Conform to the Policy?
Mini-Spec
Program
Does the Program Correspond to the Mini-Spec?
yes/no
yes/no
Both yes?
yes
Install
no
Reject

Figure 1: High-Level Overview

The term *specification* when applied to programs is usually taken to mean a general statement of *all* of the functional and/or other relevant properties of a program. To distinguish this form of specification from the more general use, the term *mini-spec* is used.

Definition 2 *A mini-spec formally states a selected subset of the functional properties of a program's behavior.*

This paper discusses the question: "Does the mini-spec conform to the policy?" Of equal concern is the correspondence between the mini-spec and the program it specifies. The scheme described in [2] proposes the use of a *filter* that will analyze a binary program and ensure that it conforms to what is stated in the mini-spec (see Figure 1). Traditionally, such an analysis has proven to be difficult. However, the assumption in [2] is that such programs should take full advantage of good software engineering techniques and need not contain the types of actions that are difficult to analyze, such as dynamic code generation, complicated computations for generating object names, and operating system manipulations. The basic premise is that reasonably engineered programs will be analyzable[2]. A reasonably engineered program is one that at least uses a structured methodology, is modular, and is written in a higher-level language. Current research described in [6] has implemented a filter program such as the one proposed in [2]. The filter approach appears promising.

An alternative method for verifying that the program conforms to the mini-spec is source code to specification correlation. The code-to-spec correlation process would have to be altered slightly since it is a one-to-one mapping between each line of code and each line of the specification. The mini-spec only states a *subset* of the program's behavior and such a mapping does not exist. However, verifying the source code against the mini-spec, as opposed to the binary, requires the existence of a trusted means for generating the binary from the source code. Without a trusted means, it would be possible to change the binary during the compilation stage.

The scope of this work is the specification of the mini-spec, development of policy, and the conformance of the mini-spec to a policy. It is assumed that mechanisms exist for verifying a program against its mini-spec, as described above. It is further assumed that once a program is verified against its mini-spec, whether by a filter program or some other means, the program and the associated mini-spec must be sealed or encapsulated in some way to prevent tampering. These issues are well understood and will not be addressed here. The APC accepts a program/mini-spec pair that has been verified and properly sealed. The next section discusses the language used for stating mini-specs and policies.

A Regular Expression Based Specification Language

This section discusses the formal language that was developed for writing mini-specs and policies. The language is based on regular expression notation. The reasons for choosing regular expressions are presented in the next section. The syntax and use of the language is provided in Section , and the limitations of the language are discussed in Section .

Why Regular Expressions?

At the level of an applications program, a system resource might correspond to a file, device, block of memory, an so on. An applications program requests system services through system calls in which a system resource is referenced by a human-readable name. A name translation mechanism converts the human-readable name to the actual page(s) on disk, memory location, etc. The name translation mechanism assumes that the supplier of the name being translated has appropriate access, leaving all access decisions to the access control mechanism, if one exists. The problem is that conventional access control mechanisms are concerned with the access between users and resources, no check is made concerning the access between programs and resources. The example provided in [5] shows how the Fortran compiler only needs access to xyz.for and xyz.obj but can easily gain access to login.com if allowed by the access control mechanism.

The APC controls the access between programs and system resources. The policy is a set of rules which states allowable program behavior. There is one rule for each type of operation under control. Each rule is a set of human-readable names of system resources accessible to that operation. For example, the "modification rule" might be a set of names of directories where modification is permitted on the system. A mini-spec is also a set of rules, one for each type of operation that must be controlled in the particular system. Thus, a program's "modification rule" would be the set of human-readable names of system resources that the program might attempt to modify.

The notion of regular expressions has long been used in the design of lexical analyzers for grouping variable names and other tokens[4]. Other uses for regular expressions include text editors, pattern matching programs, and various file-searching programs. Regular expressions are well-suited for representing a set of strings such as the set of resource names, attribute names, or system call names that can be manipulated by a program.

For ease of discussion, the remainder of this paper will discuss policies and mini-specs that have only one rule, i.e., control a single operation. It is a simple matter to extend these ideas to multiple rules.

Discussion

An *alphabet*, Σ, is a finite set of symbols. A *(formal) language*, denoted L, is a set of strings of symbols from a particular alphabet. The language Σ^* is the set of all strings over a particular alphabet Σ; thus all languages L over Σ are a subset of Σ^*. A regular expression, r, is a way of describing these languages. The notation $L(r)$ denotes the language described by r.

Let r_i be the regular expression that denotes the mini-spec for a particular operation of program i. The set of strings denoted by r_i is a finite-state language over some alphabet Σ. The language specified by r_i is denoted as

$$L(r_i) \tag{1}$$

Let p be the regular expression that denotes the policy, and $L(p)$ is the language denoted by p. Determining if the mini-spec for a given program is acceptable according to the policy of a specific system then becomes a matter of determining if the language represented by the program's mini-spec is a subset of the language denoted by the policy, for each individual rule. More formally, if

$$L(r_i) \subseteq L(p) \tag{2}$$

for each corresponding rule in the policy, then the mini-spec is acceptable according to the system's policy.

Theoretically, the answer to equation 2 is straightforward. Ultimately, we want to be able to compare the two regular expressions without having to elucidate each element in the languages denoted by the expressions. To show that this can be done, consider the following properties of regular expressions.

1. First, the languages denoted by regular expressions are precisely those languages accepted by finite automata; so $L(r_i)$ and $L(p)$ are accepted by deterministic finite automata $M(r_i)$ and $M(p)$, respectively[4, 9]. The class of languages denoted by regular expressions is closed under complementation, i.e., the complement of a language denoted by a regular expression is also a language that can be denoted by a regular expression. To show this, let $M = (Q, \Sigma, \delta, q_0, F)$[2] be a deterministic finite automaton (DFA). Let L be the language over Σ accepted by M; so $L \subseteq \Sigma^*$. Then, the complementary language, $\Sigma^* - L$, is accepted by the DFA $M' = (Q, \Sigma, \delta, q_0, Q - F)$. In other words, M and M' are the same except that the final states are opposite.
2. Second, by definition the languages denoted by regular expressions are closed under union. Therefore, given that the class of languages denoted by regular expressions are closed under complementation and union, it is simple to show that they are also closed under intersection. Let L_1 and L_2 be languages over the alphabet Σ. Then $L_1 \wedge L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Returning to equation 2, to answer the question, consider the following equation:

$$(\Sigma^* - L(p)) \wedge L_i(r) = \emptyset \tag{3}$$

Consider the language that is the complement of the language denoted by the policy. If the language denoted by the program's mini-spec, $L(r_i)$, has anything in common with the complementary language of the policy, $\Sigma^* - L(p)$, then clearly, $L(r_i)$ is not a subset of $L(p)$.

Although it can be shown theoretically that two regular expressions can be directly compared to determine if one is a subset of the other, algorithmically the problem is considered PSPACE-complete[3]. Solutions to many PSPACE-complete problems exist, and in fact, these algorithms work well when certain constraints are applied. The APC currently implements one such algorithm. The primary constraint is that the regular expressions that denote the mini-spec and the policy, must be simple enough to be processed during a reasonable processing cycle. For regular expressions that do not meet this constraint, two alternatives are available. A detailed discussion of the algorithm, and these alternatives is provided in [10].

Language Syntax and Usage

Table 1 identifies the basic operators of the language. The precedence is listed from highest to lowest with the loop operator having the highest precedence. Parenthesis are used to override the normal precedence order as the example in Table 1 shows. The first four operators listed, loop, concatenation, union, and parenthesis for grouping, are standard regular expression operators. Note, however, that the loop operator indicates $0 \leq i$ where i is limited by the maximum string length on a particular machine. Thus, the expression a^* denotes a finite language, which differs from the standard definition.

Nonterminal definitions provide user-friendliness by allowing a user to define commonly used expressions. Nonterminal definition names are 1-8 characters in length, all small letters; the definition itself is written in the operators of the language. Nonterminal definitions can be referenced via the angle brackets (< >) operator and can be embedded. The depth of macro definitions is machine dependent but it is wise to keep a limit on it. Nonterminal definitions are

[2]Where Q is the set of all states in the DFA, Σ is the input alphabet, δ represents the transition function, q_0 is the initial state, F is the set of final states, and $q_0, F \subseteq Q$.[4, 9]

Table 1: Syntax of Language

SYMBOL	MEANING	EXAMPLE
$\star$	loop - $0 \leq i$	$a^\star \Rightarrow \{\epsilon, a, aa, aaa, ...\}$
	concatenate	$ab \Rightarrow \{ab\}$
$\vert$	union	$a \mid b \Rightarrow a \cup b$; $\{a, b\}$
( )	grouping	$(a \mid b)^\star \Rightarrow \{a, b, aa, ab, ba, bb, ...\}$ $a \mid b^\star \Rightarrow \{a, b, bb, bbb, ...\}$
::=	nonterminal definition	id ::= $(a \mid b)^\star$
< >	nonterminal reference (1)	<id>$\Rightarrow$ $(a \mid b)^\star$
[ ]	series (2)	[a | b | c...]
{cwd}	current working directory	{cwd}/(a | b) $\Rightarrow$ {cwd/a,cwd/b}
{home}	home directory	{home}(a | b) $\Rightarrow$ {home/a,home/b}
files	define expression	files ::= <id>

Notes:
(1) Nonterminals are 1-8 characters, all small letters.
(2) Series can be used with nonterminal definitions.

stored in files; example nonterminal definition files, called **sysdefs** and **unixdefs**, are shown in Figure 2. A file of nonterminal definitions can be referenced via the "#include" mechanism of Unix. The square brackets operator ([ ]) is used to define a long series such as all the lowercase letters or all the digits. This operator is an implementation enhancement; parenthesis or nothing can be used to represent the same thing, i.e., $(a \mid b \mid c) \equiv a \mid b \mid c \equiv [a \mid b \mid c]$. An improvement to the current language would be to allow $[a - z]$ to indicate all the lowercase letters.

The current working directory operator {cwd} and the home directory operator {home} can be used in systems that have knowledge about filesystem location, such as Unix or Multics. In a Unix system, for example, all directories in the system would include {cwd}/, {home}/, and all other directory locations.

Policies and mini-specs are stored in files. Figure 2 shows the mini-spec for the modification operation for the `calendar` program. The last line of a mini-spec or policy file must begin with the "files" operator followed by the defines or goes into (::=) symbol as shown in the example in Figure 2. The example shows that the `calendar` program can create files in the current working directory of the form "cal" followed by a string as defined in the **unixdefs** nonterminal file. The grammar for the language just described is provided in [10].

Writing Policies and Mini-Specs for Real Programs

A mini-spec is written either during program development by a user wishing to submit a program for installation or it can be written for programs that already exist. Detailed information must be available in order to write a mini-spec for an existing program. This information might include source code, detailed design documentation, programmers notes, and test results.

Writing a policy requires knowledge about the particular threat, the system vulnerabilities, and the desired environment. Although some users may have the sophistication for writing a policy, in most cases the policy should be written by a security officer or other security personnel. Section discusses the application of the APC to the virus problem, the development of anti-viral policy, and presents results of using the APC to test for undesirable program behavior (in this case viral behavior) in 125 Unix programs.

```
sysdefs:

small            ::=     [a | b | ... | z]
large            ::=     [A | B | ... | Z]
digit            ::=     [0 | 1 | ... | 9]
special          ::=     [! | / | ... | +]

unixdefs:

atom             ::=     (<small> | <large> | <digit> | <special>)
string           ::=     <atom><atom>*
dir              ::=     /(<string>/)*

"mini-spec" for calendar:

#include "sysdefs"
#include "unixdefs"
files            ::=     /tmp/cal<string> | std(err | out)
```

Figure 2: Example Nonterminal Definitions

The Language Preprocessor

The APC first calls a preprocessor to resolve the "#include" statements, and to check the syntax of the mini-spec and the policy. The preprocessor enforces the rule that all expressions must denote a regular language (all expressions must be regular). Regular languages with an infinite number of strings are represented by the "$*$" operator in the regular expression or a cycle in the Finite State Machine. Non-regular languages do exist and cannot be represented by these constructs. For example, a language such as the one denoted by $\{a^n : n \text{ is prime}\}$ has no simple periodicity, is not regular, and cannot be represented by the constructs of regular languages[9].

The preprocessor enforces this rule by making sure that all referenced nonterminal definitions have been defined before they are referenced. A nonterminal is not defined until after the carriage return, prohibiting expressions of the form: <foo> ::= a | <foo>. This forces the use of the "$*$" operator for all loops and is sufficient to enforce that all expressions denote regular languages. The preprocessor, part of the APC, provides an error message and the line number in the file where the error occurred, when a syntax error, such as the one just described, is encountered.

Evaluation of the Language

The language for writing policies and mini-specs is based on regular expressions, which is a commonly accepted notation for representing a name space. It is a straightforward matter to use the language to represent the names of system resources manipulated by programs, such as file and device names, file attributes, environment variables, and system call names. Another application would be to *encode* behavior patterns in a regular expression, such as user profiles, using the constructs of the language.

The primary drawback to this language is that the expressions must be kept simple enough to be processed by the APC during a reasonable processing cycle. A "reasonable processing cycle" will be specific to a particular installation depending on the price, in processing time, one wishes to pay for protection from viruses. This requires some knowledge about regular expressions on the part of the individual writing the mini-spec or policy. In some cases, the mini-spec may have to be broken down into several pieces or simplified according to regular expression transformation rules.

Using the APC

The APC command is as follows:

apc name1 [name2] [-I*dir*...]

> ***name1*** **is the name of the file that contains the mini-spec;** ***name2*** **is the name of the file that contains the policy. If name2 is not supplied, the default is the system policy.**

-I*dir* **"#include" files are sought first in the current working directory, then in the directories named in the -I options.**

When the policy is not supplied by the user a default system policy can be used. This allows a user to test out the mini-spec against an individual policy before submitting it to the system administrator for installation. It also allows the user to have an individual policy that is more stringent than the system policy. For example, suppose the system policy allows modifying of any files in any user directory. If a user does not wish to allow modification of the home directory, then the user can write an individual policy that only allows modifications to files not in the user's home directory. The user can then use the APC, supplying the user-specific policy, when deciding whether to execute new programs. The APC returns a message indicating whether or not the program is acceptable according to the policy.

Application of the APC to the Virus Problem

This section discusses the application of the APC to the virus problem. All experiments were conducted under SunOS. The distinguishing characteristic of a computer virus is its ability to infect other programs by *modifying* them to include a copy of the virus. Although there are other behaviors of programs that can be controlled to prohibit viral activity, all of the policies discussed here focus on the modification operation, specifically, the modification of files and directories. All of the policies contain a single rule which specifies the directory and file names where modification is allowed. All of the mini-specs also contain one rule which specifies the directory and file names that the associated program could attempt to modify.

Test Suite

All of the programs in sections 1 & 6 of the Unix Reference Manual[11] were examined for possible inclusion in the test suite[3]. These programs include editors, compilers, game programs, printing programs, and other basic utility programs available to normal users. The modification behavior of each program were studied by reading the Unix manual pages, looking at source code when available, and talking with Unix developers when necessary. In some cases, the modification activity of a program could not be adequately identified due to the lack of sufficient information. Such programs could not be included in the test suite. A total of 125 programs comprise the test suite.

Many of the programs in the test suite had the same modification behavior. A total of twenty-three unique mini-specs were written to represent the 125 programs in the test suite. Three additional mini-specs were written to simulate programs infected with a real virus. The reason for including "infected" programs was to show whether each policy prevented infected programs from being accepted. All twenty-six mini-specs were tested against each identified policy. The details of the program study, the mini-specs, and the programs they represent, are presented in detail in [10].

Policies

The development of the anti-viral policies was approached from opposite angles. On the one hand, normal user activity was identified and several policies were developed to allow that behavior. On the other hand, several viruses were identified and policies were written to prohibit their behavior. The basic methodology was to develop policies that allowed normal user behavior and prohibited abnormal (viral) behavior.

[3]All of the programs in these sections of the manual are available to normal users. For purposes of these experiments, programs which require special privileges were not considered.

Policies for Normal User Activity

The first policy considered is a loose policy that allows modification to any directory and any file on the system. The reason for including such a policy is to show the operation of the system with respect to viruses when no restriction is placed on allowable modification activity. This policy is comparable to no policy and all programs in the test suite, including the three "infected" programs, were accepted. At the opposite end of the spectrum of policies is a tight policy that only allows modifications to the standard output and standard error devices. This policy is included to represent a policy that allows very little modification activity. A policy such as this effectively keeps all viruses out of the system, it does not accept any editors, compilers, or programs to manipulate files.

Policy 3 allows modification to files in specific directories: /dev/, /tmp/,{cwd} or {home} This policy does not allow modifications to any other directory, nor does it allow modifications to any subdirectory of these directories. A total of 50% of the programs in the test suite, are accepted by this policy. More importantly, this policy successfully rejects all three "infected" programs. Policy 4 is more restrictive and only allows modifications to the current working directory; this does not include files in subdirectories of the current working directory. Only 42% of the programs in the test suite are accepted by this policy. All three "infected" programs are also rejected. Policy 5 is the opposite of policy 4 in that modifications are allow to any file located anywhere in the system *except* the current working directory. A total of 59% of programs in the test suite were accepted by policy 5, however, two of the "infected" programs were also accepted. Policy 6 is also very restrictive; this policy only allows modification to objects located in the temporary directory /tmp/. Although this policy correctly rejects the three infected programs, it only accepts programs 42% of the programs in the test suite.

Of all the policies in this section, policy 3 which allows modifications to all four specific directories appears to be the best policy in that it accepts that largest percentage of programs. None of the policies accept any compilers or editors.

Policies to Prevent Specific Viruses

Four Unix viruses are identified in this section. The details of each virus are not presented for security reasons. Instead, each virus is described in terms of the name space of directories and/or files that it modifies.

The Murray Unix Virus infects Unix shell programs. Murray looks for shell programs to infect in the user's bin/ directory and the current working directory. Murray also creates and modifies several files in the current working directory that start with a ".". Since shell programs are not identifiable by their name, the policy is written to prohibit any modifications to the user's /bin/ directory or current working directory (modifications to subdirectories of the current working directory are permitted). Furthermore, this policy only allows modification to files in the home directory that do not start with a ".". Modifications to other files in other locations in the system are permitted. This policy accepts the same set of programs that were accepted by policy 5 - the policy that does not allow modification to the current working directory. Although this policy rejects the mini-spec "infected" with the Murray virus, the other two "infected" programs are accepted.

To simulate the IBM Christmas Tree virus in a Unix environment, policy 8 was written. This virus is not technically a virus since it doesn't copy itself to another program, i.e., the virus doesn't *infect* other programs. Instead, this virus, or worm, sends a copy of itself to all of the electronic addresses of all the users listed in the victim's address alias file. To stop the spread, policy 8 prohibits modification of the mail spool directories, the location where all outgoing mail is queued until it is sent out of the system. This policy accepts 61% of the programs in the test suite. However, it restricts the proper usage of the mail programs so that mail cannot be sent out of the system by anyone. Also, this policy does not reject the other two "infected" mini-specs.

The virus described in [1] is a general virus that searches for any executable file and appends itself to the executable. Since this virus can modify any file any where in the system, policy 9

Table 2: Policy Acceptance Rate

Policy	Name	Mini-Specs Accepted	Total Programs Represented	% of Programs Accepted	Infected Programs Accepted
1	Loose	*all*	125	100%	yes
2	Tight	1	48	38%	no
3	Specific Directory	1,7,10-12, 14,18,20	63	50%	no
4	{cwd}	1,11,14,20	52	42%	no
5	not {cwd}	1-7,9,10,12, 15,17-19,22,23	74	59%	yes
6	/tmp	1,10,12	52	42%	no
7	Anti-Murray	same as 5	74	59%	yes
8	Anti-Xmas	1-7,9-12, 14,15, 17-20,23	76	61%	yes
9	Anti-Generic	1,7,18	55	44%	no
10	Anti-Worm	1-6,9,10,12, 14,15,18-20	64	51%	no
11	Combo	1,18	49	39%	no

prohibits all modifications except to the devices. Such a policy is very restrictive and, although it successfully prohibits all viruses, it allows only 44% of programs.

Policy 10 is intended to prohibit the Internet Worm. The Internet worm modified many things on the system. Most important were the sockets that it wrote to in order to transfer the worm from the host machine to the victim machine. The worm used unnamed sockets which makes it impossible to use this scheme to prohibit writing to sockets. The worm also created files beginning with the letter "x" which it later deleted in an attempt to hide itself. Prohibiting modification to files whose name begins with the letter "x" would halt the Internet worm but it would be a simple matter to re-write the worm to use some other letter. Policy 10 does prohibit the "infected" mini-spec which represents a program that is carrying the Internet Worm and it accepts 56% of all useful programs. This policy also successfully rejects the other "infected" programs. However, this policy would be very simple to circumvent. A second generation of this virus could choose a different letter or randomly select a letter other than "x". In this way the virus would be accepted by the policy.

Policy 11 was developed by using the union operator of the language and combining policies 7, 8, 9, & 10. The reason for including such a policy is to experiment with a single policy for all viruses vs. a policy for each individual virus. This policy does successfully reject all three "infected" programs, but it only accepts 39% of all programs. Also, it is easy to see that this policy doesn't prohibit *all* viruses, just those described here.

Evaluation of Empirical Results

Table 2 shows the acceptance rate for each policy just described. Column 1 identifies the policy by number, column 2 lists the number of the mini-specs that were accepted, column 3 shows the total number of programs represented by the accepted mini-specs. Column 4 indicates the percentage of the 125 programs in the test suite that were accepted by each policy. The last column indicates if any of the "infected" mini-specs were accepted.

Eleven total policies were identified: 6 policies allow normal user behavior while 5 policies prevent a specific virus(es). Policies 1, 5, 7, and 8 accepted one or more of the "infected" mini-specs; mini-specs that were included to represent programs infected with a virus. Policies 10 & 11 are considered weak policies as it would be very simple to create a virus to circumvent these policies. Policy 3, which allows modifications only to specific directories (/dev/, /tmp/, {cwd}, {home}) accepts the largest number of programs. In general, policies based on normal user behavior accept a larger percentage of programs, especially most necessary programs; those based on specific viruses are easily circumvented.

None of the policies that are effective against viruses accept any editors, compilers, linkers, or other programs considered necessary for normal user operation. This leads to the conclusion that basing the policy on the modification behavior of programs, although an important activity to control for virus prevention, by itself is inadequate. There are two reasons for this. First, the nature of Unix programs is that they either modify *only* standard error and/or standard out (39% of programs in the test suite), or they modify any file in any directory (34% of programs in the test suite). This results in mini-specs that specify program behavior as "all or nothing" for 72% of the programs in the test suite. Clearly, this approach would be more effective if the modification characteristics of Unix programs were restricted. The question is: could this be done easily without a great deal of impact on users?

The second reason is the unavailability of user input. This approach is a static, preventative mechanism, it is applied once, prior to program installation. The mini-spec attempts to capture the potential dynamic behavior of a program but because of its static nature, this results in many programs being rejected at installation time that could operate within the confines of the policy at run-time. For example, suppose the policy states that the only modifications allowed are to /tmp/ and files can have any name as long as they do not begin with the letter "a". If the mini-spec for a particular program modifies files of any name in /tmp/ the program would not be accepted for installation. If the mechanism were applied at run-time instead, the program might execute within the confines of the policy, i.e., not modify any files that begin with the letter "a". Thus, the unavailability of user input results in policies that appear overly restrictive.

The modification behavior of a program is the most obvious characteristic of a computer virus which is why it was chosen as the behavior of focus for this study. However, although important for virus control, policies based on this behavior are not restrictive enough, too restrictive, or can easily be circumvented. Alternative behaviors to be considered include system call patterns, modification of file attributes, modification of environment variables.

Lastly, the fact that the language does not contain a construct for intersection, and especially complementation, results in policies that are more complex than if these operators had been available. Policy 8, which is intended to prevent the Murray virus, is an example of this. Since there is no way to directly express policies such as "anywhere but /bin/", policies tend to be overly restrictive. An alternative would be to specify all the file names and directories that *cannot* be modified in the policy. Then, if the APC determines that the mini-spec is *not* a subset of the policy then the mini-spec would be considered acceptable.

Evaluation of the Overall Approach

This research has shown that the proposed mechanism can keep viruses out of a system and still accept a percentage of most necessary programs; it is a feasible and practical approach. It has further been shown that controlling the modification of files is an important behavior to control for virus prevention, but results in policies that are too restrictive, not restrictive enough or results in policies that can easily be circumvented. The APC clearly separates the policy from the mechanism that enforces the policy. This will allow future studies to investigate alternative behaviors for virus prevention and control.

On the negative side, the complex expressions that result due to the lack of the intersection and complementation operators of the language, sometimes result in lengthy execution times. However, the approach suggested in the previous section, i.e., testing for whether a given mini-spec is *not* a member of the policy's language, may provide a simple solution to this problem. Also, the inability to capture run-time user input results in a greater number of programs being rejected. A possible solution to this problem is to move the mechanism into the run-time environment that would not require a mini-spec and could operate on the actual file or directory name; a much simpler check would need to be made in this case. The next section discusses future research.

Conclusions and Future Work

The investigation of alternative program behaviors is an obvious extension to this work since it would not require any additional implementation. Since the mechanism and policy are clearly separated, the study of alternative behaviors simply means supplying a new test suite. Several possibilities for other behaviors to investigate include system call patterns, file attribute modifications and environment variables. In the case of system call patterns, a possible approach is to encode the patterns of modification system calls in a regular expression using the constructs of the language. The same is true for file attributes or environment variables although specific names could also be used in the same way as the current study. In any case, developing a new test suite is not a simple matter. In order to provide a useful test suite, a large set of programs should be investigated and studied. As this research has shown, policies based on normal user activity are the most effective.

To accommodate user input, this mechanism could be extended to represent user behavior. For example, user behavior could also be encoded in a regular expression using the constructs of the language. The union operator of the language could then be used to combine a particular user's behavior with that of a specific program. The combined specification could then be checked against the policy. This would involve a study of user behavior, particularly user modification behaviors. The advantage of this approach would be the information regarding each specific user rather than just the program's behavior. This could also be classified by class of users or group of users.

Another obvious extension of this work is to implement the mechanism as a run-time mechanism and to run experiments with test suites in both the static, prevention mode and the dynamic, detection mode. This would provide information regarding prevention vs. detection of viruses.

Lastly, extending this work to different types of systems, such as a DOS or Macintosh system, would be a useful project. In both cases, neither system takes advantage of the memory protection features of the hardware, allowing programs to modify any location in the system. The mechanism described here, especially if implemented as a run-time mechanism, could be used to restrict the modification activity of programs despite the failure to use memory protection. Since many of the DOS viruses encountered directly modify memory addresses, an appropriate behavior to study might be the actual calls for modification rather than the directory and file names as was done in this study.

This research has implemented a mechanism for comparing program behavior against a policy that states allowable program behavior. This approach has been applied to the virus problem and shown to be a practical and feasible approach for preventing computer viruses. This mechanism does not have a high impact on performance, and does not result in inconvenience to users. When used with a filter program such as one described in [2, 6], it does not rely on some number of humans to act in a trustworthy fashion. Most importantly, this approach clearly separates the policy from the mechanism that enforces the policy. In this way a variety of policies and program behaviors can be studied and tested using the APC.

References

[1] F. Cohen. Computer Viruses. In *Proceedings of the 7th National Computer Security Conference*, pages 240–263, 1984.

[2] S. Crocker and M. Pozzo. A Verification-Based Virus Filter. In *IEEE Symposium on Research in Security and Privacy*, 1989.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractibility A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, San Francisco, CA, 1979.

[4] J.E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Inc., Reading,MA, 1979.

[5] P.A. Karger. Limiting the Damage Potential of Discretionary Trojan Horses. In *IEEE Symposium on Security and Privacy*, 1987.

[6] P. Kerchen, R. Lo, J. Crossley, G. Elkinbard, K. Levitt, and R. Olsson. Static Analysis Virus Detection Tools for Unix Systems. In *13th National Computer Security Conference*, October 1990.

[7] N. Lai and T.E. Gray. Strengthening Discretionary Access Controls to Inhibit Trojan Horses and Computer Viruses. In *Proceedings of the Summer 1988 USENIX Conference*, 1988.

[8] R. Levin, E.Cohen, W. Corwin, F. Pollack, and W. Wulf. Policy/Mechanism separation in Hydra. In *Proceedings of the Fifth Symposium on Operating Systems Principles*, November 1975. University of Texas at Austin.

[9] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1981.

[10] M.M. Pozzo. *Towards Computer Virus Prevention*. PhD thesis, University of California, Los Angeles, 1990.

[11] Sun Microsystems, Sun Release 3.2. *Commands Reference Manual*, 1986. Mountain View, CA.

IMPROVEMENT OF DATA PROCESSING SECURITY BY MEANS OF FAULT TOLERANCE

Gilles Trouessin* Yves Deswarte* Jean-Charles Fabre* Brian Randell**

* LAAS-CNRS & INRIA
7, Avenue du Colonel Roche
31077 Toulouse Cedex – France

** Computing Laboratory, The University
Newcastle upon Tyne
NE1 7RU – United Kingdom

Abstract

This paper discusses various different solutions to the problem of reliable processing of confidential information. One of the major difficulties of this problem comes from the fact that conventional techniques for achieving reliability, on the one hand, and security on the other, tend to be in opposition to each other. The different solutions presented here have been classified in three distinct types: two are related to classical security techniques (protection, and encryption) and the third is a new technique, the fragmentation-redundancy-scattering technique, which it is claimed demonstrates a potentially advantageous unified approach to the provision of reliability and security, based on fault tolerance. Finally, a qualitative comparison of these solutions is given, taking into account both dependability, openness and performance criteria.

Keywords: secure architectures, integrity, reliability/availability, protection, encryption, fragmentation.

1. Introduction

In this paper we concern ourselves with the provision of high reliability and availability, and the preservation of data confidentiality, in large scale distributed systems, such as ones based on workstations connected over one or more high speed LANs.

1.1. Problem statement

Dependability, a generic concept – defined as *the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers* – may be viewed w.r.t. different *properties* [8] and so enables the definition of a number of different dependability **attributes**, including: **availability** (w.r.t. *readiness for usage*), **reliability** (w.r.t. *continuity of service*), **safety** (w.r.t. *avoidance of catastrophic consequences on the environment*), **security** (w.r.t. *prevention of unauthorized access and/or handling of information, i.e., provision of data integrity and confidentiality*).

Some of these attributes (reliability/availability and security) are often considered separately because the techniques used to achieve them are usually perceived as being mutually antagonistic. Firstly, *reliability* and *availability* are generally achieved by incorporating mechanisms for tolerating any faults (especially accidental faults) that occur, or that remain despite attempts at fault prevention during the system design process. These techniques will of necessity involve space and/or time redundancy; they can easily take advantage of a distributed computing architecture by means of replicated computation using sets of untrusted[1] (or fallible) processors. Secondly, *security features* are generally achieved by means of fault prevention mechanisms (w.r.t. intentional faults, such as intrusions) whereby critical applications are implemented on physically and/or logically protected computers. Such protection is usually based on the *TCB* (Trusted Computing Base) or *NTCB* (Network Trusted Computing Base) concepts [17] [18].

From the above we see that what can be termed an antagonism between reliability/availability and security arises in at least the two following ways [7]: (i) *accidental-fault tolerance* (by means of replication) increases the number of potential access points to confidential information and thus can reduce the effectiveness of the protection techniques; (ii) *intrusion prevention* (by means of a local *TCB* or a *NTCB* partition) can suffer from the fact that one cannot justifiably rely either on a single *TCB* (which forms a classical "single point of failure"), or on the local *TCB/NTCB* partition of each computer inter-connected to the network.

To be adequately realistic, a solution dealing with this antagonism must, we believe, take into account the following two requirements: (i) *trusted area reduction*, by which we mean that the security provided by a potential

1 Here we use the term *trusted* component to mean one that is assumed to be highly reliable and available, and impervious to intrusions (i.e., not to be a source of deliberate faults), in its intended environment.

solution should depend on as small as possible a *trusted area*, because it is impossible to place confidence on all of the processors in the network; (ii) *openness*, by which we mean that a potential solution must not be (excessively) software and/or hardware dependent, but instead allow implementation over a network of heterogeneous systems. The former requirement, regarding *trusted area reduction*, would also contribute to the latter one, regarding *openness*, since untrusted processors belonging to same critical application would thus not be security-dependent.

1.2. Reliable processing of confidential information

Let us consider the problem of *reliable processing of confidential information* as involving a combination of the three following features *a), b)* and *c)*:

a) Simple processing (Fig. 1a): processing *(P)* is applied to a set of input data *(D)* in order to obtain a set of output results *(R)*. Both areas are shaded to denote the fact that neither the processor (the lower area) nor the environment containing the I/O (input/output) devices and which provides the inputs and accepts the outputs (the upper area) are trusted.

b) Reliable processing (Fig. 1b): the redundant execution of *P* (by means of processor replication) in order to provide data integrity for *D* and *R*, and reliable processing of *P*.

c) Confidential processing (Fig. 1c): in this, and indeed all cases where confidentiality is required, input is provided from, and output is delivered back to, a trusted area. Neither of the two regions of Fig. 1c is shaded; this is to indicate that *P* is executed, and its I/O prepared/received, securely (in similarly trusted areas), in order to preserve the confidentiality of *D*, *R* and (perhaps) *P*.

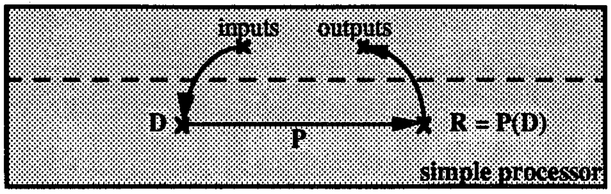


Figure 1a: Simple processing

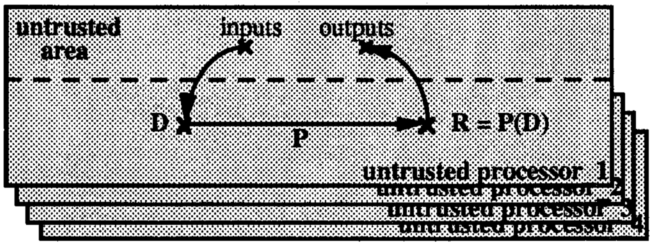


Figure 1b: Reliable processing

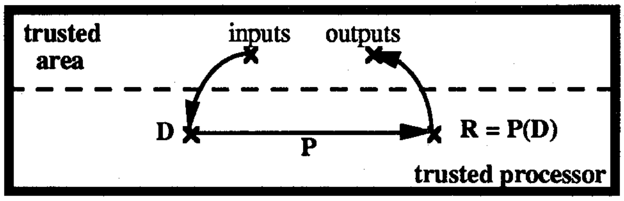


Figure 1c: Confidential processing

2. Achieving Combined Reliability and Security

2.1. Approach 1: Protection

This first approach is based on a classical security technique, *protection*, an intrusion-prevention technique which is based on forecasting and preventing, as far as possible, the different intrusions that could damage overall system security. This technique may be implemented by either of two different solutions: *1) centralized protection* or *2) local protection*. In each case, replication is also employed, in order to add both processing reliability and data integrity.

2.1.1. Solution 1.1: Centralized protection and replication

This first solution (Fig. 2) is in fact the logical combination of the features (reliability and confidentiality) represented in Figs. 1b and 1c. As in all cases where confidentiality is required, I/O operations, for each given user, are performed in a trusted area using a similarly trusted processor. However, the processing is replicated and executed by trusted processors that all belong to the same trusted area as that where the data is provided and the results received by the user. Solution 1.1 can be developed on the basis of a centralized *TCB* as recommended in the *Orange Book* [17], using a specific architecture, i.e., a fault-tolerant computer system, such as Tandem or Stratus systems.

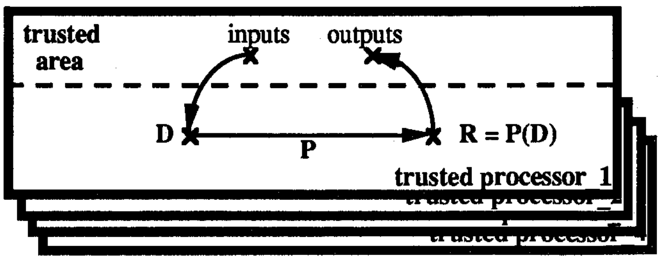


Figure 2: Solution 1.1 – Centralized protection and replication

There are two possibilities for preserving the confidentiality of data whilst it traverses the medium used for inter-processor communication, depending on whether or not the medium is considered as part of the trusted area. In the latter case, confidentiality preservation of the whole critical application is based on the encrypting of all communications between processors. In either case, the *trusted area reduction requirement*, and thus also the *openness requirement*, are not adequately met, since all the processors (together, one can be sure, with significant portions of their operating systems), and perhaps the communication medium, are considered as part of the trusted area.

Solution 1.1 is thus in practice perhaps well suited for very specific highly critical applications such as some types of military computation but does not fit well with general-purpose applications which may invoke remote processors and use several distinct networks.

2.1.2. Solution 1.2: Local protection and replication

This second solution (Fig. 3) is in fact a network generalization of the previous solution. I/O operations are performed in a trusted area located on a special trusted processor. Normal processing is still replicated but it is now accomplished in an untrusted area, on processors which are in general untrusted. Each of these processors is however protected by means of a local *TCB* and a *NTCB* partition as recommended in the *Red Book* [18].

An *Authentication—Access Control* scheme (*AAC* and *AAC'*, in Fig. 3) is needed between the special trusted processor and the other processors involved in the critical application, in order to ensure the overall security of the application.

There is only one possibility for the preservation of the confidentiality of the communication medium since processing is executed in the untrusted area: the medium must be considered as part of the untrusted area and all communications between the different processors must thus be encrypted.

Several hardware and/or software implementations of this solution have been developed, for example: the *Distributed Secure System* [2] [11], the *LOCK co-processor* [12] in connection with the *SDNS* project [15], *Secure Sun OS* [16].

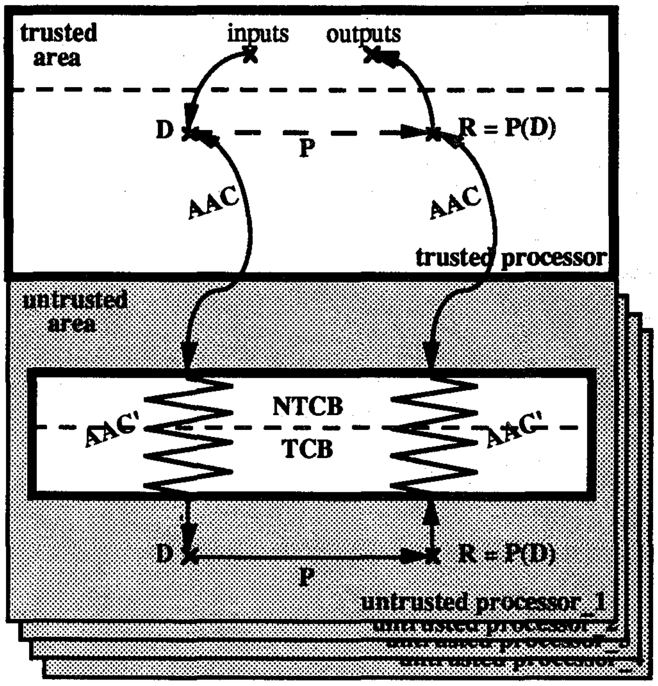


Figure 3: Solution 1.2 – Local protection and replication

With Solution 1.2, we can see that the *trusted area reduction requirement* is partially respected: (i) *respected*, since each processor involved in executing the critical application is now considered to be untrusted, and to be situated in an untrusted environment area; (ii) *partially*, because each of these untrusted processors must be protected by a local *TCB* and *NTCB* component, which are each in fact a local (albeit perhaps small) trusted software and/or hardware mechanism operating in an untrusted environment. However, this means that the mechanism should therefore be made tamper-proof, so that it cannot be opened without destroying its content. Such tamper-proof devices also need to possess a master key in order to communicate securely with other such devices in the untrusted area; and in practice, must be small and essentially maintenance-free.

The other requirement, openness, is not respected because each implementation of this solution requires the help of a *TCB/NTCB* partition to enforce security on the different processors of the network. This is the main drawback of Solution 1.2, particularly where the *TCB* or *NTCB* component is merely software running on the otherwise untrusted processor, because it is very difficult to protect the component from and by something as complex as an operating system, (e.g., Unix in the *LOCK/ix* project). However when the *NTCB* is in special-purpose hardware, monitoring all communications to and from the untrusted processor (as in the *DSS* project), its task, and that of making it tamper-proof, are more readily achievable. Anyway, in all cases, if any of the local *TCB/NTCB* components are corrupted or replaced by Trojan Horses, all the others are threatened so that the security of the whole network is compromised and the confidentiality of the critical application lost.

2.2. Approach 2: Encryption

This second approach is based on another classical security technique, *encryption*, which is a well established technique for preserving the confidentiality of communications and file archiving. It can be used for preserving the confidentiality of information processing in two different ways: *1) homomorphic encryption* or *2) black-box encryption.* In each case, replication is also used, in order to provide both processing reliability and data integrity.

2.2.1. Solution 2.1: Homomorphic encryption and replication

With this solution (Fig. 4), a user's I/O operations are as always performed in a trusted area, and reliability features are obtained by means of processing replication, again in an untrusted area, but in a encrypted way. In the one trusted area, a special trusted processor transforms the data set *(D)* and the processing *(P)* by means of a specific kind of encryption technique *(C)* into an encrypted data set *(D')* and encrypted processing *(P')*.

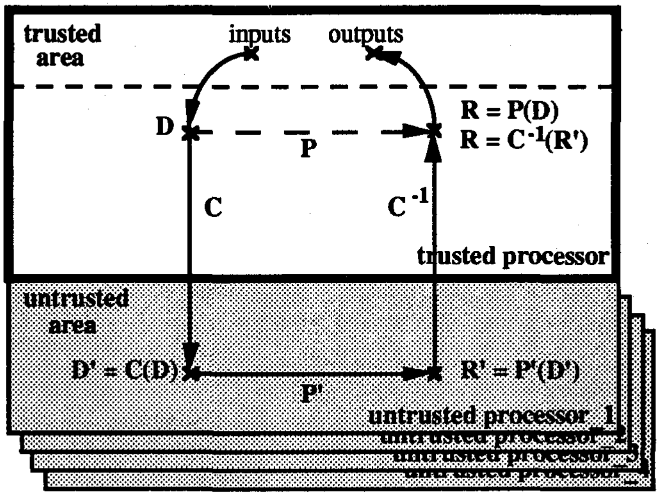


Figure 4: Solution 2.1 –
Homomorphic encryption and replication

Only certain types of encryption, called *privacy homomorphisms* [1] [10] [13], are suitable for such transformations. However, when *C* is of such a type, *P'* can be securely accomplished in the untrusted area, by untrusted processors. Encrypted results *(R')* obtained in the untrusted area can then be de-crypted (C^{-1}) in the trusted area to obtain results in clear *(R)*:

- *D* thus has an image *D'* according to *C*: $D'=C(D)$;
- *P* also has its own "image" *P'* depending on both *C* and *P* features: *P' is a function of (C,P)*;
- *R'* is thus an image of *D'* with *P'*: $R'=P'(D')$.

With Solution 2.1 communication confidentiality is directly preserved by means of encryption and no additional techniques are required for this purpose.

But a restriction must be observed in implementing any scheme based on Solution 2.1. If an intruder can access the encrypted value of any arbitrary constant and if the comparison operator is available then usage of a privacy homomorphism is no longer secure. This is because the intruder can use a simple binary search strategy to discover the encrypted value of each data item of the whole data set *D* [10]. However in some particular cases (where there is no need for a comparison operator) Solution 2.1 is valid [1] [13]; but these cases are very limited (very specific banking transactions, for example) and thus this approach cannot be considered as providing a general solution.

Because of the above restriction, we can say that Solution 2.1 partially respects the *openness requirement* since processing is securely executed only in some particular cases (if *C* is a privacy homomorphism and if *P* does not provide the comparison operator). However, we can say that Solution 2.1 respects the *trusted area reduction requirement* perfectly, since processing is completely executed by untrusted processors, without any need for trusted devices in the untrusted area.

2.2.2. Solution 2.2: Black-box encryption and replication

This solution (Fig. 5) exhibits some common features with the previous solution: I/O operations are performed in the trusted area and processing in the untrusted area, reliability is obtained by replication and confidentiality by encryption. However, homomorphic encryption is replaced by *black-box encryption.* In fact, processing is apparently executed in encrypted form: $R'=P'(D')$, since only encrypted data *D'*, encrypted processing *P'* and encrypted results *R'* can be observed in the untrusted area.

In reality, processing is executed in clear inside a trusted "black box" associated with each untrusted processor. This solution involves three steps:

- encrypted input data *(D')* is received and de-crypted with C^{-1}: $D=C^{-1}(D')$;
- normal processing *P* is executed in order to obtain results *R*: $R=P(D)$;
- results *R* are encrypted with *C* in order to obtain *R'* that can be sent out of the black box securely: $R'=C(R)$.

The trusted black box thus contains a decrypting-processor, a small size memory, a processor and an encrypting-processor. To be really secure, it must be tamper-proof (as described in Section 2.1.2 above).

However, Solution 2.2 suffers from several major drawbacks [7], though the first three listed below are essentially similar to those possessed by Solution 1.2:

- *protection against a Trojan-horse black box:* in order to be qualified as *trusted*, it must not be possible to replace the black box by a *Trojan-horse black box* during its operational life (leave alone during initial installation);
- *management of encrypted addresses:* all data received by the trusted black box, such as addresses, are encrypted and are thus more difficult to decode and use;
- *management of communication keys:* one (or several) master cryptographic key(s) is(are) required in order to allow secure communications, which increases the management complexity of key distribution and use;
- *increase of local memory space:* for management of encrypted addresses or communication keys and local data storage, thus increasing the local memory space required for the black box whereas it ideally should, as mentioned previously, be small.

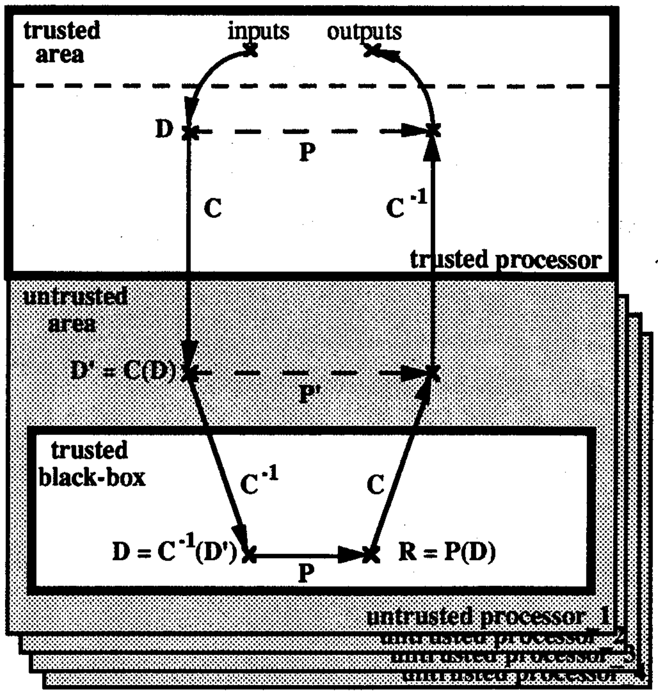


Figure 5: Solution 2.2 –
Black-box encryption and replication

Because of these drawbacks, we can say that though Solution 2.2 is feasible, like Solution 1.2 it does not meet the *openness requirement*, because the security in the untrusted area is really hardware- and software-dependent (i.e. black box dependent), and it does not meet the *trusted area reduction requirement* very well, since a trusted device (the trusted black box) must be installed essentially in each processor.

2.3. Approach 3: Fragmentation-Scattering

This third approach is based on what can be termed a "unified fault tolerance" technique, the *Fragmentation-Redundancy-Scattering (FRS)* technique, since it provides, in a single mechanism, means of tolerating both accidental and intentional faults, and hence of providing both reliability *and* confidentiality of data and its processing. Fragmentation involves defining fragments of information so as to ensure that, once isolated into physically separate processors, each fragment is of little value to a potential intruder due to the lack of significant information content in any one processor. (In principle such fragmentation can either be achieved at the programming language level, where it can take advantage of programmer-defined data structuring, or at the operating system level, where it is based on machine-level data types, such as bytes, words and/or pages. Particularly in the former case there is the possibility of requiring, and making use of, programmer-supplied constraints indicating which data items it would be especially undesirable for an intruder to be able to correlate.) Such fragments are then replicated, and the replicated fragments scattered across a (preferably large) number of processors.

FRS has been developed and successfully demonstrated in the context of a secure file archiving system [5] [6] and in the course of research into security management [3] [4]. In the processing context [7] [19] the approach relies on the correct execution of a majority of a set of copies of each of a number of program fragments with their corresponding data fragments, these fragments being widely distributed across a number of untrusted processors. Research to date on the application of *FRS* to processing[1] has resulted in the devising of two rather different implementation schemes: *1) fragmentation-scattering and replication* or *2) fragmentation-scattering and threshold.*

[1] The *FRS* technique applied to processing is called *Fragmented Data Processing (FDP)*. Some actual examples of this *FDP* technique are presented in the Appendix to this paper.

2.3.1. Solution 3.1: Fragmentation-scattering and replication

With this solution (Fig. 6), I/O operations are again performed in the trusted area, reliability features are again obtained by means of processing replication in the untrusted area, but in a fragmented fashion. In the trusted area, the trusted processor transforms the data set *(D)* by means of a set of projections, or data-fragmentation functions, $F = \{f_1, f_2, \ldots, f_n\}$, into a set $D = \{d_1, d_2, \ldots, d_n\}$ of data fragments. Similarly, processing *(P)* is transformed by means of a set of projections, or program-code fragmentation functions, $G = \{g_1, g_2, \ldots, g_n\}$, into a set $P = \{p_1, p_2, \ldots, p_n\}$ of program-code fragments. A critical application is thus split into *n* distinct program fragments, each of which consists of a data fragment d_i and a program-code fragment p_i, as follows:

- d_i is the image of *D* by projection f_i: $d_i=f_i(D)$;
- p_i is the image of *P* by projection g_i: $d_i=g_i(D)$;
- r_i is the image of d_i by processing p_i: $r_i=p_i(d_i)$.

Results *R* can only be reassembled on the trusted processor because each untrusted processor does not have enough information to permit such re-assembly: perhaps just a single program fragment (one data-fragment d_i, one program code-fragment p_i) and thus one result fragment r_i, for a given application. In practice, several program fragments could however be mapped on the same physical processor, provided that they do not in sum reveal any significant information.

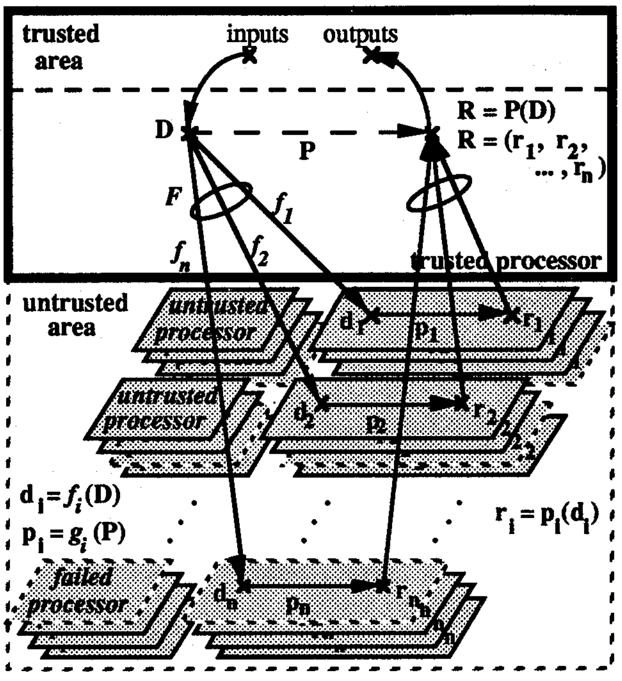


Figure 6: Solution 3.1 – Fragmentation-scattering and replication

Solution 3.1 possesses two main beneficial features:

- *different classes of fragmentation functions (f_i and g_i) can be defined:* security depends on the way the fragmentation functions (f_i and g_i) are chosen: for a given critical application different classes of fragmentation functions are possible (data and/or program-code driven) and different fragmentation strategies are also possible, thus allowing different security features to be obtained (data and/or program-code confidentiality preservation) [19] [20];
- *security does not depend on f_i or g_i confidentiality:* security does not rely on a potential intruder being ignorant of the semantics of the fragmentation functions (f_i or g_i).

A potentially major drawback of Solution 3.1 is that it might be expensive, in terms both of performance and program development effort. The major issues involved are as follows:

- *additional memory space overheads:* the memory space overheads due to replication are exactly the same as for any other solution using replication; those due to fragmentation come from the fact that there can be an overlapping of the data fragments d_i derived from *D*. In such a case and if these overheads are important, then another fragmentation strategy (based on a larger, but then admittedly perhaps less secure, fragmentation granularity) might be adopted;
- *increased number of processors:* this is unavoidable, but in the introduction of this paper it was indicated that we are assuming the basic global environment of the problem of *reliable processing of confidential information* in a distributed computing environment. In many such systems, for example university computing networks, a large number of processors can be idle very often [9]; in such cases the provision of reliability/availability by means of processing replication and of security by means of fragmentation-scattering can together take advantage of such unused processing power;
- *communication overheads:* since many more processors are required than with the two previous approaches, much more communication traffic may be induced; for example by data fragment exchanges between distinct program fragments. In such cases, and if communication overheads are significant, this again, might motivate the adoption of another fragmentation strategy, with larger fragmentation granularity;
- *development effort:* if significant development effort is needed to apply the technique to each separate application this would constitute the major cost of this approach, which would probably only be justifiable on very critical

applications which were thereafter to receive extensive use. To date, investigations have been somewhat application-specific, but the prospect of application-independent methods of using the technique is now being considered.

From the above, and from experiments to date, we claim that solution 3.1 respects the *openness requirement*; we believe that most types of critical application can be fragmented, at least to a degree, largely because a wide range of fragmentation possibilities are offered by means of the different fragmentation classes and strategies. The *trusted area reduction requirement* is certainly respected since no trusted software and/or hardware components need be situated in untrusted areas. In particular, the fragmentation functions (f_i and g_i), though they are used in the trusted area and are the basis of security, are not confidential and could actually be held in untrusted areas.

2.3.2. Solution 3.2: Fragmentation-scattering and threshold schemes

This solution (Fig. 7) has some points in common with the previous one: I/O operations are performed in the trusted area, processing in the untrusted area, and confidentiality requirements obtained by means of fragmentation-scattering. But reliability/availability are now obtained by using so-called threshold schemes [14] applied to processing, instead of using processing replication. With the threshold technique, at least s (the threshold number) shadows of a given secret are needed to reconstitute the secret and less than s shadows do not give any information about this secret. Like error correcting codes, the threshold scheme technique thus imposes some redundancy in order to tolerate accidental but also intentional faults (especially intrusions w.r.t. data confidentiality and integrity), and during processing.

In the trusted area, the special trusted processor transforms the data set (D) by means of a specific set of projections, taking into account the threshold scheme, $F^s = \{f_1^s, f_2^s, \dots, f_m^s\}$, into a set of data fragments, $D^s = \{d_1^s, d_2^s, \dots, d_m^s\}$. Similarly, processing (P) is transformed by means of a specific set of projections, $G^s = \{g_1^s, g_2^s, \dots, g_m^s\}$, into a set of program-code fragments: $P^s = \{p_1^s, p_2^s, \dots, p_m^s\}$. A critical application is thus split into $m>n$ distinct program fragments (each consisting of a data fragment d_i^s and a program-code fragment p_i^s) as follows:

- d_i^s is the image of D by projection f_i^s: $d_i^s=f_i^s(D)$;
- p_i^s is the image of P by projection g_i^s: $d_i^s=g_i^s(D)$;
- r_i^s is thus the image of d_i^s by p_i^s: $r_i^s=p_i^s(d_i^s)$.

As with Solution 3.1, R can only be reconstituted on the trusted processor because each untrusted processor does not have enough information, possessing in principal just one data and one program-code fragment for a given application.

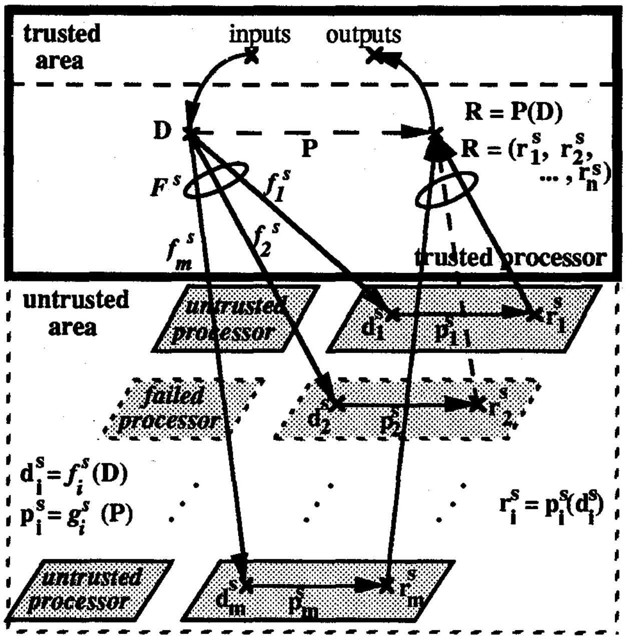


Figure 7: Solution 3.2 – Fragmentation-scattering and threshold schemes

Solution 3.2 possesses many of the advantages and disadvantages of Solution 3.1 - however, it allows reduction of the number of required processors to m, instead of the $n.r$ needed by the previous solution (if r is the replication level) and thus reduces various types of overhead, including communication overheads. But its main drawback is that not all threshold schema are suitable. With Shamir's threshold schema based on polynomials [14] for example, a restricted set of polynomials must be chosen: addition is conserved with this kind of processing, but multiplication becomes quickly expensive (high degree polynomials are required in order to perform multiplications securely) and comparison can be used only if the restricted polynomial set offers the total order property (any shadow of any secret must be comparable and respect the total order property imposed by the secrets).

Thus we can see that this solution meets the *trusted area reduction requirement* fully and that just a restricted set of threshold schemes can be used (high degree and ordered polynomials). Two main problems must also be considered: (i) the security of the fragmentation projections (F^s and G^s) must remain as strong as the (proved) security of the threshold scheme technique; (ii) the cost of these fragmentation projections, i.e., in terms of development effort and required execution time, must not be too important compared to the execution time of the program-code fragments (P^s).

3. Qualitative Comparison

Different comparison criteria must be considered, *dependability*, *openness* and *performance criteria*, to provide a qualitative comparison (Table 1) of the different solutions presented in the previous section.

3.1. Dependability, openness and performance criteria

As explained in the introduction of this paper, *reliable processing of confidential information* is assumed to be concerned with different dependability criteria *(dc)*, i.e., goals and requirements, and one openness criterion *(oc)*:

- *dc_1* – **reliability:** data processing reliability (and availability) and data integrity;
- *dc_2* – **confidentiality preservation (security):** preservation of data (and perhaps processing) confidentiality;
- *dc_3* – **trusted area reduction:** non excessive security-dependence on trusted areas;
- *oc_1* – **openness:** non excessive security-dependence on specific software or hardware.

Five performance criteria *(pc)* are considered, since valid solutions must not involve excessive performance overheads:

- *pc_1* – **number of processors:** the number of processors required, not counting those needed for redundancy purposes (see *pc_2*);
- *pc_2* – **redundancy overheads:** the number of processors required for redundancy purposes;
- *pc_3* – **number of messages:** the number of messages induced by the total set of processors;
- *pc_4* – **memory size:** the local (to each processor) memory space required for the solution implementation;
- *pc_5* – **system connectivity:** the number of inter-connection links required between the different processors.

As yet we do not have extensive experimental data concerning the performance and costs of *FRS*, when applied as here to processing, as distinct from file archiving and security management. Therefore Table 1 gives our subjective comparison of the six distinct solutions presented in this paper, taking into account the above dependability and performance criteria. Five values, corresponding respectively to the following qualitative scale, are used to characterize the extent to which the different criteria are satisfied:

- the five values are: ——, –, ≡, + and ++;
- the corresponding qualitative scale is: *very unfavourable*, *unfavourable*, *no influence*, *favourable* and *very favourable*.

Table 1:
Qualitative comparison of the different solutions with respect to dependability, openness and performance criteria

	dependability			open-ness	performance				
	rel.	security							
	dc_1	*dc_2*	*dc_3*	*oc_1*	*pc_1*	*pc_2*	*pc_3*	*pc_4*	*pc_5*
	reliability/availability/integrity	confidentiality preservation	trusted area reduction	openness	number of processors	redundancy overheads	number of messages	memory size	system connectivity
Approach 1: **protection** *Sol. 1.1:* centralized protection + replication	≡	–	——	——	++	≡	≡	≡	≡
Sol. 1.2: local protection + replication	≡	–	≡	–	++	≡	≡	≡	≡
Approach 2: **encryption** *Sol. 2.1:* homomorphic encryption + replication	≡	–	++	≡	+	≡	≡	≡	≡
Sol. 2.2: black-box encryption + replication	≡	–	≡	–	+	≡	≡	≡	≡
Approach 3: **fragmentation-scattering** *Sol. 3.1:* fragmentation-scattering + replication	++	++	++	++	——	≡	——	+	–
Sol. 3.2: fragmentation-scattering + threshold schemes	+	+	++	++	≡	+	≡	+	≡

3.2. Comparison results

Table 1 shows that in our opinion Approach 3 leads to better results w.r.t. the dependability (and openness) criteria: this is due to the fact that it is a unified concept providing both accidental- and intentional-fault tolerance. Approach 1 and Approach 2 globally present better results w.r.t. performance criteria *pc_1*, *pc_2*, and *pc_3*; this is due to the fact individual processing replicas (*D*, *P* and *R*) are not split over different processors, as is the case with Approach 3.

It should be noted that local (but not global) memory overheads are not expected to be significant with Approach 3; this is because more processors are required for almost the same global memory occupation and each processor then needs a smaller local memory space.

From this analysis and the judgements expressed in the Table, we conclude that all of the approaches described here are in principle capable of providing solutions, suitable for at least some situations, to the problem of *reliable processing of confidential information*. Each approach, however, has some weaknesses:

- *Approach 1:* poor reduction of the trusted area, and either poor security openness (Solution 1.1) or poor confidentiality preservation (Solution 1.2, when based on OS-enforced software partitioning);
- *Approach 2:* poor confidentiality preservation, and either poor security openness (Solution 2.1: operator restriction) or poor reduction of the trusted area (Solution 2.2);
- *Approach 3:* though adequate w.r.t. our dependability requirements this approach is somewhat poor w.r.t. performance, a situation which we believe can be ameliorated by exploiting the large range of possible fragmentation strategies already identified for this new technique.

4. Concluding Remarks

The originality and potential attractiveness of the fragmentation-scattering approach is that it is provides a unified means of achieving both reliability and security. At this stage, much remains to be discovered about its real advantages and disadvantages, w.r.t. the pre-existing combined techniques against which we have compared it, but we believe the analysis presented above shows that it has considerable promise as a new means of providing fault tolerance against both accidental faults and intentional faults such as intrusions.

Acknowledgements

The authors wish to thank very much Jean-Claude Laprie and David Powell for their earlier basic ideas about the *Fragmentation-Redundancy-Scattering* approach for ensuring high reliability/availability and security, Jean-Michel Fray for his contribution to the concept of *Fragmented Data Processing* for reliable processing of confidential information, and the reviewers of the submitted version of this paper for their constructive comments. This work was partially supported by DRET under convention no. 88.34.051.00.470.75.01 and the PDCS (Predictably Dependable Computing Systems) project no. 3092 of the European ESPRIT programme.

References

[1] N. Ahituv, Y. Lapid, S. Neumann, *Processing Encrypted Data*, Comm. of the ACM, Vol. 30(9), Sept. 1987, pp. 777-780.

[2] D.H. Barnes, R. MacDonald. *A Practical Distributed Secure System*, Proc. NEOS'85 (Networks and Electronic Office Systems), London – United Kingdom, IERE, 1985, pp. 83-91.

[3] L. Blain, Y. Deswarte, *An Intrusion-Tolerant Security Server for an Open Distributed Computing System*, Proc. European Symposium On Research In Computer Security, ESORICS'90, Toulouse – France, Oct. 24-26, 1990, Pub. AFCET, ISBN 2-9036778-9, pp. 97-104.

[4] Y. Deswarte, L. Blain, J.-C. Fabre, *Intrusion Tolerance in Distributed System*, Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland – California, May 20-22, 1991, pp. 110-121.

[5] J.-M. Fray, Y. Deswarte, D. Powell, *Intrusion Tolerance Using Fine-Grain Fragmentation-Scattering*, Proc. 1986 IEEE Symposium on Security and Privacy, Oakland – California, April 7-9, 1986, pp. 194-201.

[6] J.-M. Fray, Y. Deswarte, D. Powell, *A Secure Distributed File System based on Intrusion Tolerance*, Proc. 4th International Conference on Computer Security, IFIP/Sec'86, Monte Carlo, Dec. 2-4, 1986, pp. 407-416.

[7] J.-M. Fray, J.-C. Fabre, *Fragmented Data Processing: an Approach to Secure and Reliable Processing in Distributed Computing Systems*, Proc. 1st IFIP International Working Conference on Dependable Computing for Critical Applications, Santa Barbara - California, Aug. 23-25, 1989, pp. 131-137, published in *Dependable Computing for Critical Applications*, Ed. A. Avižienis, J.-C. Laprie, Springer Verlag 1991, Vol. 4, ISBN 3-211-82249-6 & 0-387-82249-6, pp. 323-343.

[8] J.-C. Laprie, *A Unifying Concept for Reliable Computing and Fault-Tolerance*, in *Dependability of Resilient Computers*, Ed. T. Anderson, Blackwell Scientific Publications, 1989.

[9] D.A. Nichols, *Using Idle Nodes in a Shared Computing Environment*, Proc. 11th ACM Symposium on Operating Systems Principles, Austin – Texas, Nov. 1987, pp. 5-12.

[10] R.L. Rivest, L. Adleman, M. Dertouzos, *On Data Banks and Privacy Homomorphisms*, in *Foundations of Secure Computation*, Ed. R.A. Demillo, D.D. Dobkin, A.K. Jones, R.J. Lipton, Academic Press, 1978, pp. 169-179.

[11] J.M. Rushby, B. Randell, *A Distributed Secure System*, IEEE Computer, Vol. 16(7), 1983, pp. 55-67.

[12] S.O. Saydjari, J.M. Beckman, J.R. Leaman, *Locking Computers Securely*, Proc. 10th National Computer Security Conference, Sept. 1987, pp. 129-141.

[13] J. Seberry, J. Pieprzyk, *Cryptography: An Introduction to Computer Security*, Ed. R.P. Brent, Prentice Hall, 1989, 375 p.

[14] A. Shamir, *How to share a secret*, Communications of the ACM, Vol. 22(11), Nov. 1979, pp. 612-613.

[15] E.R. Sheehan, *Access Control within SDNS*, Proc. 10th National Computer Security Conf., Sept. 1987, pp. 165-171.

[16] B. Taylor, D. Goldberg, *Secure Networking in the Sun Environment*, USENIX Association Summer Conference Proc., Atlanta – Georgia, 1986, pp. 28-37.

[17] TCSEC, Department of Defense Standard, *Trusted Computer System Evaluation Criteria*, Department Of Defense, DOD 5200.28-STD, Dec. 1985, 121 p.

[18] TNI, National Computer Security Center, *Trusted Network Interpretation of TCSEC*, National Computer Security Center, NCSC.TG.005, July 1987, 278 p.

[19] G. Trouessin, J.-C. Fabre, Y. Deswarte, *Reliable Processing of Confidential Information*, Proc. 7th International Conference on Information Security, IFIP/Sec'91, Brighton – United Kingdom, May 15-17, 1991, pp. 210-221.

[20] G. Trouessin, *Quantitative Evaluation of Confidentiality by Entropy Calculation*, Proc. 4th Computer Security Foundations Workshop, Franconia – New Hampshire, June 18-20, 1991.

Appendix

Depending on the way that fragmentation is performed, it is possible to define different classes of fragmentation techniques, which are concisely described below and more detailed in [19]. In addition, fragmentation granularity is another parameter that can be considered in the choice of a given fragmentation class.

The first class of fragmentation technique relies on fine grain fragmentation (bit or small group of bits), and is in effect a sort of bit-slicing technique: each data item within the whole data structure is split into f fragments of b bits. Thus each of the individual processors has only a local view of the data structure, i.e., b bits out of $f.b$ bits. By this means, the global value of each data item and thus its semantics can be effectively hidden. The code must then be transformed into a set of code "fragments", each of which has been modified so as to work appropriately with the corresponding fragmented data. Actually, this is not really code fragmentation since the code is simply slightly transformed, and the original program's semantics is unlikely to be effectively disguised from an intruder. The main interest of this class of fragmentation technique is that it allows a quantitative evaluation of confidentiality preservation by means of entropy calculation [20], but its main drawback, due to its restriction to fine granularity only, is that the *openness requirement* is not respected.

A second class can be defined when applied at the module level. Each program fragment (data- and code-fragment) corresponds to a single entity in the whole program structure (an instruction block, a program module or a library function) delimited by breaks in the code sequence. Each such fragment is then replicated in r distinct copies scattered over the network. Actually, this class of fragmentation technique is opposite to the previous one because the code is first fragmented in connection with its structure (this is particularly interesting in the case of modular programming languages or block-structured languages) and then the whole data structure is consistently fragmented (with respect to the first code fragmentation). This means that each code fragment will be associated with its own local variables and this implies also that global variables must be fragmented and shared by several distinct fragments.

From the above two classes of fragmentation technique we can derive a third one. As with the first class, this third class takes into account the data structure in order to apply a first step of fragmentation. This first step is well suited to the preservation of data confidentiality, since its aim is to cut some of the semantic links in the tree that could otherwise be built with the main semantic links of the data structure. And secondly, as with the second class, it is applied at the module level and thus similarly relies on the program structure, so is a technique which is well suited to preservation of code confidentiality. This third class thus is used at the programming language level, and is a compromise between fragmentation at the variable and program module levels. If fine grain fragmentation is required for efficient confidentiality preservation then data and code fragmentation can be applied in alternation to get an overall fragmentation which depends on both data and code structures.

Information Security: Can Ethics Make a Difference?

Corey D. Schou
Associate Professor
Department of Computer Information Systems
Idaho State University

John A. Kilpatrick
Associate Professor
Department of Management
Idaho State University

ABSTRACT

Information is a vital organizational asset that affects ongoing decision making. It has a finite life span therefore if it is delayed in its distribution, it has reduced value; if a proper user fails to have access, it has no value.

The objective of attempts to secure the organizational information system is to see that unauthorized use is not possible, that destructive viruses are not introduced, and that unauthorized study and alteration of records and files does not occur during the distribution of data and information throughout the organization while guaranteeing that proper users have easy access to their information. Are these objectives strictly technical problems, or is it possible and appropriate to broaden the scope to include the ethical issues that are raised as the security system is developed and installed? The argument in this paper is that it is both appropriate and necessary to consider the broader issues.

INTRODUCTION

Information is the lifeblood of an organization that over the years has become recognized as an asset. Although determining the value of this asset from an accounting standpoint is difficult, it should be protected like any other. One of the dominant characteristics facing any firm attempting to become and to stay competitive is the dependence on information processing that relies on computers and computer software. In this paper we attempt to address many of the ethical issues facing managers in organizations as they attempt to cope with the complexity and cost of acquiring, integrating and securing information systems in the workplace. In large organizations, this task is assigned to an Information Resource Manager who is responsible for all aspects of information processing from data entry to the Executive Information System[1]. This manager 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 the organization.

In the process of performing this task, the manager must balance two competing objectives that are for all practical purposes antithetical. The first is that of ease of access to information to meet the requisite variety needs of decision makers within a system[2]. The second is that of maintaining security, confidentiality and privacy of organizational information assets.

1. Schou, Corey D., "Computer Security: Training Needs for Managers," *Data Management*, Auerbach, September, 1990.

Frequently, this process is viewed as a technical problem rather than the more complex socio-technical problem that should address some of the following issues:

- Whose rights are to be considered?
- To what extent are these rights in conflict?
- What are the responsibilities of the information specialists?
- How honest and trusting are the members of the user community? In what sense do they represent a 'community'? What are the implications, if any, of their holding certain interests in common?
- How trusting ought they to be? What is implied in the use of the term *ought*? Of the term *trust*?

These socio-technical problems are fundamental ethical issues. These issues may or may not be legal issues. The manager should be aware that which is legal is not necessarily a logical equivalence of either ethical or right.[3]

QUESTIONS OF PURPOSE AND VALUE

Since there is a documented body of law that governs portions of our behavior and a cult of technology which asserts that it can make our electronic information systems invulnerable to external penetration, we tend to rely upon it. To complete the protection of our information assets, we must develop an awareness of value systems. This development must be more than another set of rules and regulations that dictate how we should behave. They should be, on the other hand, an internalized set of behaviors. We should ensure that rules and technology do not become the sole focus of our security activities. These are destined to fail of their own weight in the long term. John LaCarré? in one of his novels makes the point about the impact of technology on human activity by stating:

George Smiley: "You've made technique a way of life. Like a whore, technique replacing love."[4]

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 remains for living. As this result suggests, it is easy to become obsessed with the tools and in the process to forget the purpose of the tools.

Although information security systems are adequate from the successful system control, they do not always take into account the corresponding human impact and implications. This brings us to the question - What is the role played by the values that members of a community hold that form the choices made and the ends toward which those choices are directed? 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 that individuals make, the goals that are pursued, and the priorities that are established. They affect both the means that are selected and the ends toward which efforts are directed.

2. Beer, Stafford, *The Brain of the Firm*, John Wiley & Sons, New York, NY, 1981.

3. Richards, T., Schou, C.D. & Fites, P.E. "Information Systems Security Laws and Legislation," in *Information Technology Resources Utilization and Management: Issues and Trends*, Idea Group, Harrisburg, Pa., 1990.

4. LaCarré, John, *Tinker, Tailor, Soldier, Spy*, Doubleday, New York, NY, 1986.

ETHICAL SYSTEMS

What are the purposes of computer information systems? Information systems are organizational mechanisms that collect data and distribute information. Frequently these systems rely on electronic devices such as computers; however, the 'office boy' carrying a scrap of paper to the file drawer also meets this definition.

Some systems relate to governmental objectives (e.g., national defense, collection of revenue, monitoring of international trade), some to business purposes and needs (e.g., efficiency and competitiveness), but all must relate at some level to social needs and values. For example, one might argue that a fundamental value is respect for the rights of others. Another might be that the overall objective is a better quality of life for all members of the community.

There are several ways of identifying and deciding ethical issues. One of the most common Judeo - Christian ways of categorizing these approaches is the rules *Vs.* consequences criteria. The first argues that our actions should be guided by general rules or principles: do not harm; tell the truth; do not steal; have respect for persons as 'ends in themselves.' The second argues that we should assess the *rightness* of an action or decision by the consequences that will likely result. Most commonly the second approach identifies some value or values, and measures an action by the extent to which these values are or are not enhanced, or whether progress is made toward certain goals, such as a better life for all. From a practical standpoint it may be recognized that, for most people, over a span of time and in different situations, both approaches will be used. That is, in general some ethical rule may seem appropriate but under extreme circumstances exceptions to the rule or principle will appear ethically acceptable because of the likely consequences.

ETHICS AND INFORMATION SYSTEMS

For an information system to function effectively and efficiently, there must be a free flow of data and information among all participants. In the ideal situation, there would be no barriers to this flow; this would improve the probability that 'perfect information' is in the hands of the decision makers. Of course, for this to occur, there would have to be perfect confidence and trust within the organization.

Confidence and Trust

Information - adequate, relevant, timely, understandable - is a precondition of an efficient and free society. Yet it is a means to power . . . Therefore, the rights to create property in information, to withhold, to disclose, to determine when and how disseminated are critical[5].

In this section we are interested in the ethical issues involving the creation, control, use, abuse, dissemination, protection, manipulation or alteration, examination and destruction of information and its attendant data in computer systems. In order for the above information activities to take place efficiently and legitimately, there must be some minimal level of trust and confidence in the systems which handle the information. Is it also necessary for there to be some minimal level of trust among and between the various users of the system?

Assuming that such a level is necessary, what are the preconditions in order for this confidence and trust to exist? It appears clear that first there must be a proven and recognized history of dependability, both within the firm and with similar systems.

5. Behrman, Jack, "Information Disclosure, the Right to Know and the Right to Lie'" in Behrman, *Essays on Ethics in Business and the Profession*, PrenticeHall, Englewood Cliffs, NJ., 1988, 79.

By raising these issues in the context of the firm's culture or atmosphere, one ethical principle is implied: that there must be respect for persons and certain property rights. This falls within the first approach identified above, which argues for the assessment of choices in light of certain ethical principles or rules. Actions which result in intrusion, examination, alteration or destruction of information belonging to others might be judged as morally wrong because they violate the principle of respect for persons. The second approach, that of looking at the consequences of an action, might suggest that in order for a community to meet the needs of its members, individuals within the community must be able to have some confidence in systems of communication. According to this view, it could be argued that actions that unduly interfere with the smooth operation of information and communication systems, or that diminish the confidence and trust in these systems, should be judged as unethical.

Definitions

As a starting point for determining ways of evaluating actions, it is appropriate to construct several definitions. The term *legitimate* is fundamental to the notion of balancing rights and responsibilities. For the purposes of this paper, it is argued that for an action or behavior affecting an information system to be legitimate, it must aid in the achievement of one or more objectives of the system without unduly interfering with progress toward other accepted objectives. The definition can be applied to the ethical management of information. One objective of the system is to provide information that is without deception and is understandable, timely, relevant, complete and appropriate to the user. Upon examination, it can be seen that this definition suggests both the practical and ethical elements of managing computer information systems.

SPECIFIC CONCERNS RELATING TO THE DESIGN OF SECURE SYSTEMS

Those involved in the design of a secure information system must be aware of the conflicting rights, responsibilities and needs of system users and professionals, and of the implications of some of these conflicts. Some paradoxical assertions may serve to illustrate:

- For people to have trust in an information system, the manager must trust no one.
- Systems which are truly trustworthy must use control processes that inhibit use.

Another way of putting the problem, as Clifford Stoll suggests in his book, *The Cuckoo's Egg*,[6] is that as administrative controls are added to ensure trustworthiness, the system becomes more difficult to use. This means that the people for whom the system is designed end up finding another, less trustworthy but more easily accessible system to use. The term administrative controls refers to those policies and procedures imposed by a manager that 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 managers desire. Generally this means, in ways that advance organizational objectives through fixed procedures. 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 to set standards for the entire firm. Frequently, however, it also refers to the need to regulate behavior when it is perceived that:

6. Stoll, Clifford A., *The Cuckoo's Egg*, Doubleday, New York, NY, 1989.

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 individuals always coincided 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 individual malevolent intent.

Finally, there are those instances where it is in 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 excellent metaphor, it is the "tragedy of the commons"[7] . 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. One lesson for members of the community is that, unless they are willing to eliminate all cooperative efforts, the exercise of some moral restraint by each individual is necessary.

EXAMPLES OF ETHICAL ISSUES CONFRONTED IN ORGANIZATIONS

As long as the information system consists of 'office boys' carrying paper from place to place, the problems are less complex. If he takes something home - he has stolen - he is wrong. However, when the organization begins to rely on electronic means, this issue becomes more clouded. The same individual can take or send electronic images of the same information without overtly changing it. (After all, what is the value of a simple '0' or '1'.)The following are examples of some problems that are uniquely electronic.

Pirated Software

One of the more obvious and most prevalent problem deals with the use of pirated software. The temptations are obvious and the risk of disclosure is slight. Why then the concern? There are several ethical issues here, but perhaps the overriding one is that of the failure to recognize intellectual property.

As with many ethical concerns, one can arrange many positions along a continuum. In this instance, one can take an extreme individualist or ethical egoist position, and argue that pirating another's software is not a big issue, and is useful for financially strapped organizations. Further, one can argue that it is the responsibility of the developer to take measures to limit the ease of pirating. In any case, is it stealing if the property isn't gone?

At the other end is the argument that:

a) there are rights that are being violated while copying;
b) that no community can exist that refuses to acknowledge and protect the rights of its members; and

7. Hardin, G., "The Tragedy of the Commons," *Science*, 162, December 1968, 1243-1248.

c) that progress will be limited unless there is some incentive for individuals to develop tools that will prove useful in solving the problems of the community.

The manager then must address the issue of whether to allow - profit from - the pirating of another's intellectual creation, or, if the policy is to ensure that this does not occur within the business, what policies will be required to ensure that it does not occur.

Criminal Entry

Even if one has problems recognizing intellectual property, physical property is easier to define. This situation is analogous to the problem of the 'office boy' If someone breaks your physical lock, or physically enters your premises, there is little question about 'right'.

However, the problem of unwarranted entry into proprietary electronic information systems with criminal intent is more complex. Using technological means, each firm will obviously wish to ensure that its own system will not be so penetrated. What of information gained either inadvertently or through the wizardry of an employee who also happens to enjoy the challenge of breaking into another institution's information systems? Since any technological means of protection may be compromised by 'wizardry' it is important that one engender an atmosphere of ™correctness' within the organization.

Computer Surveillance & Employee Records

In a 1931 speech, George Bernard Shaw observed:

> An American has no sense of privacy. He does not know what it means. There is no such thing in the country.

At the time he may have been correct; however, the American society has matured during the last sixty years. Even though Supreme Court candidates have been unable to define the absolute nature of the rights of privacy on a constitutional basis, most Americans believe that they have a vested right of privacy based on the Fourth Amendment to the Constitution[8] . This for the most part protects us from our government.

Computerization of information systems has made the communication and dissemination of information about companies and individuals an accepted procedure. The issue of computer surveillance and employee records involves questions about the uses of databases that may involve invasion of privacy, either the customer's or employee's, and employee monitoring in the workplace. This latter involves the inclusion of a piece of software in the information system which monitors and times or otherwise measures the activities of operators. Is this a legitimate managerial exercise of administrative control, or is it an unwarranted intrusion into the employee's privacy? Put another way, should the firm legitimately be concerned only with the quantity and quality of the employee's activities, or may it also surreptitiously monitor the employee on a minute by minute basis? Questions of the impact on morale aside, how far may the manager extend his or her control over the activities of the employee? The sensitivity of this issue becomes more acute when the ability to control is magnified or enhanced by the computer's capacities. One other issue in this category involves the cross-reading or matching across information systems of employee or customer records. Again, the issue involves the right to privacy of employ-

8. Amendment IV Right of search and seizure regulated. The right of people to be secure in their persons, houses, papers, and effects against unreasonable searches and seizures and not be violated, and no warrants shall issue, but upon probable cause, supported by oath or affirmation and particularly describing the place to be searched, the persons or things to be seized.

ees and customers. Formerly, this may have been an ethical concern only in firm's large enough to have extensive databases. Today, even small organizations may have the computer capacity, or have access to databases that give the firm the capacity to intrude into the privacy of employees and customers.

The owner/manager of a small firm, then, is faced with many of the same ethical dilemmas that managers in large firms face. Dealing with the issues may be more difficult in that the small firm manager must be all things to all people, with little time for contemplating the complexities of the ethics of the computer age.

Gaming

An example of an issue of interest with perhaps least clear cut ethical stands is the use of company facilities for office games, such as 'rotisserie baseball' and 'fantasy hockey'. Employees face an ethical choice over the extent to which such 'enlivening' activities can legitimately be carried on during company time.

Managers face the need to balance productivity interests with maintaining a livable working environment that is not so rigid and controlling that the quality of work life drives off good employees.

SOURCES OF GUIDELINES AND CODES OF ETHICS

There are a not less than five organizations that have chosen to address directly the ethical issues posed by the rapid expansion of information technology they are:

- British Computer Society,
- Institute of Electrical and Electronic Engineers,
- Institute for Certification of Computer Professionals,
- CCP and
- The Data Processing Management Association.

The Data Processing Management Association (DPMA) has developed a code of ethics and a separate 'Standards of Conduct.'[9]

Standards of Conduct

These standards are derived from the code of ethics and are specific statements of behavior that no true professional will violate. Excerpts are provided below, as examples of ethical guidelines that are being developed by industry professionals:

In recognition of my obligation to management I shall:

- Not misuse the authority entrusted to me.
- Not misrepresent or withhold information concerning the capabilities of equipment, software or systems.

In recognition of my obligation to my fellow members and the profession I shall:

- Be honest in all my professional relationships.
- Not use or take credit for the work of others without specific acknowledgement and authorization.

In recognition of my obligation to society I shall:

- Protect the privacy and confidentiality of all information entrusted to me.
- To the best of my ability, insure that the products of my work are used in a socially responsible way.
- Never misrepresent or withhold information that is germane to a problem or situation of public concern nor will I allow any such known information to remain unchallenged.

9. *DPMA Code of Ethics*, Data Processing Management Association, 505 Bussie Highway, Park Ridge IL.

- Not use knowledge of a confidential or personal nature in any unauthorized manner or to achieve personal gain.

In recognition of my obligation to my employer I shall:

- Avoid conflict of interest and insure that my employer is aware of any potential conflict.
- Protect the privacy and confidentiality of all information entrusted to me.
- Not exploit the weakness of an information system for personal gain or personal satisfaction.

SUMMARY

If, due to security restrictions, an information system cannot disseminate its contents to those who need access, it fails. Technology alone does not solve the problem. It is a human problem.

It is of benefit to each user if everyone exercises discretion, judgment and professional respect for other's rights in the use of a computer information system. Each knows then that the system can be 'trusted.' 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 an individual does desire to access 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 an individual user to violate that trust. Self-restraint thus can be seen as a prerequisite for any activity requiring trust.

The violation of the trust, if discovered, necessitates a higher level of administrative control, new restrictions placed on access, and that additional procedural processes be installed. The violations have caused a reduction in the efficiency and effectiveness of the system. A fundamental consideration for the manager, 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 firm and individuals affected.

This dilemma serves to highlight the ethical considerations facing the manger. For smaller organizations, it is further complicated by resource limitations, both financial and human. 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 issues raised here:

- If people will not exercise moral restraint, systems will develop controls for protection;
- The controls for protection will prove burdensome and inefficient;
- The systems will fail;
- 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;
- The systems will fail to be secure.

"INFORMATION SECURITY RISK ANALYSIS AND RISK MANAGEMENT: WHICH APPROACH ?"

Prof. Jan H.P. Eloff
Karin P. Badenhorst
Department of Computer Science, Rand Afrikaans University
P.O Box 524, Johannesburg, 2000, Republic of South Africa
FACSIMILE: 27-11-489-2138   TELEPHONE: 27-11-489-2842

ABSTRACT

The IRR research model as proposed in this paper can be seen as an important first phase of a research process, aimed at formulating a new approach to risk analysis, risk assessment and risk management within the information technology environment. Over the past decade, considerable resources and efforts have gone into developing and automating risk analysis methods, in an attempt to make risk analysis more easily applicable and as a whole more successful. This resulted in a large number of automated techniques, methods and packages being currently available on the information security software market. The perspective the authors took in preparing this paper was to address the question "Which approach combined with underlying business philosophies and business technologies ?" This opposes the question usually asked by organisations, namely "Which package ?"

KEYWORDS: risk analysis; risk assessment; risk management; risk resolution; information security methodology; information technology; environmental risk assessment; financial risk management.

0. INTRODUCTION

Information risk assessment is a vital business management task.[15] General managers usually have a high appreciation for risks relating to the continuation of their business. However, in practice the authors observed that a considerable amount of apprehension are still felt by many managers of organisations regarding the application of information technology risk analysis.

A fundamental issue of information security is rooted in the conflict between efficiency and control. This is exactly why risk analysis and risk management is such an important part of an overall information security function within an organisation. The objective of a risk analysis and risk management exercise is to find the optimum balance between efficiency, control and the cost of such control for an organisation. As management problems addressed in information security are usually more economic and politically based than technical, this should provide management with sufficient motive to conduct a risk analysis exercise.

Management approaches problems with subjective rather than objective solutions. On the other hand, risk analysis technology has traditionally focussed on objective or deterministic issues. Effective management should use risk analysis and risk management techniques in their proper role - as a management tool, not as a substitute for good judgement.[15]

The process of risk analysis and risk management in the context of information technology is concerned, firstly with the identification and measurement of risks related to information technology, and secondly with the control and minimisation of such risks. For the remainder of this paper we will refer to the process of information technology risk analysis and risk management as Information Risk Resolution (IRR).

1. IRR: INFORMATION RISK RESOLUTION

IRR has for a number of years been applied in the computer related industry without substansive rate of success. Research done since 1983 identified an increasing dissatisfaction with previously and currently available IRR methods and approaches.[16] Based on current literature and practical experience the authors came to some conclusions regarding IRR methods:

- It should be comprehensive in terms of handling all aspects of an IRR process, so that one does not have to apply more than one method and/or tool to accomplish meaningful results.[16] On the other hand it is the authors' opinion that IRR should not be so elaborate, that it defeats the other objective, namely to make it simpler and less time-consuming.

- It should also be comprehensive in terms of information security. It must be flexible enough to cover all aspects of computer and information security, as well as the interdependencies amongst those aspects.[20] The authors believe that IRR should be addressed from a multi-dimensional as well as a multi-disciplinary perspective. The multi-disciplinary concept stems from functional computer security levels (hardware, software, personnel, program controls, etc.). The interrelationships between tasks within these functional security levels (such as identifying threats related to the physical computer room, and determining the cost of logical access controls) constitute a multi-dimensional character.[1]

- The authors are of opinion that the assessment of risk is a functional rather than a financial issue. Evident from the application of IRR in organisations is the fact that IRR is usually performed by functions such as Audit (internal and/or external), and Finance.

- A method must be flexible enough to be "calibrated" to an environment. This also holds true for the maintainability of such method - it must respond to changes in the nature of a company's business.[9] The authors agree that it must be possible to customise an ideal IRR methodology for specific types of industry and varying management styles.

- A more qualitative and less quantitative method seemed preferable to refinements of existing qauntitative methods.[16] The reasoning behind this is that quantitative figures can be misleading, because the fact that a figure is exact, does not necessarily mean that the assumptions on which the figure is based, are reliable.[25]

- Methods should reduce the amount of time, cost and overhead of performing an IRR. Such methods should therefor preferably be automated.[16]

- A risk analysis program should not be some arcane program applied on an ad hoc basis, when some unusual expense needs to be cost-justified. It should rather be an integral part of the business system.[7]

- It is very important that the method must be credible and trusted to the people that apply it, or those that rely on the results thereof.[9]

- IRR should always be applied within the perspective and as part of a comprehensive information security methodology. The likelihood of success will be greatly enhanced if it is not seen as a stand alone exercise.[2]

Consequently a number of automated packages were developed so as to make IRR "hopefully" more successfull. This resulted in a large number of automated techniques, methods and packages being currently available on the software market.

CRAMM for example makes use of qualitative scalar techniques, whereas LRAM's quantification of risk is based on formal Bayesian probability theory and decision models.[20,14] MARION assesses business risks quantitatively and/or qualitatively making use of sophisticated mathematical and statistical principles.[19] RANK-IT is based on the so-called Delphi techniques, where expert opinion plays a major role in the assessment of risk.[12] LAVA, on the other hand, makes use of binary tree concepts - it uses hierarchical disaggregation structures to link questionaires with event trees for vulnerability assessment.[26] It further makes use of linguistic algebra and fuzzy set theory. It is clear that divergent enabling techniques and approaches are used from one method to the next.

We are somewhat concerned that the root of the problem has not been addressed. To add to this problem, we are of opinion that information security risk analysis is a controversial issue amongst information security specialists, auditors, information technology managers, insurance surveyors and line managers, who respectively approaches IRR from an individual biased point of view. This statement is based on practical experience in major organisations as well as the overall impression gathered at international computer security congresses.[22]

The question to be addressed should not be "Which package to use?" but instead "Which approach combined with underlying business philosophies and business technologies ?" This problem brought us to the idea of "What makes Risk Analysis successfull in the general business or public context ?" The authors then decided to attempt a new approach to IRR through investigation of existing risk analysis techniques and methods which normally turns out successfully in business functions outside the scope of information technology.

The above mentioned formulates the scope for the remainder of this paper.

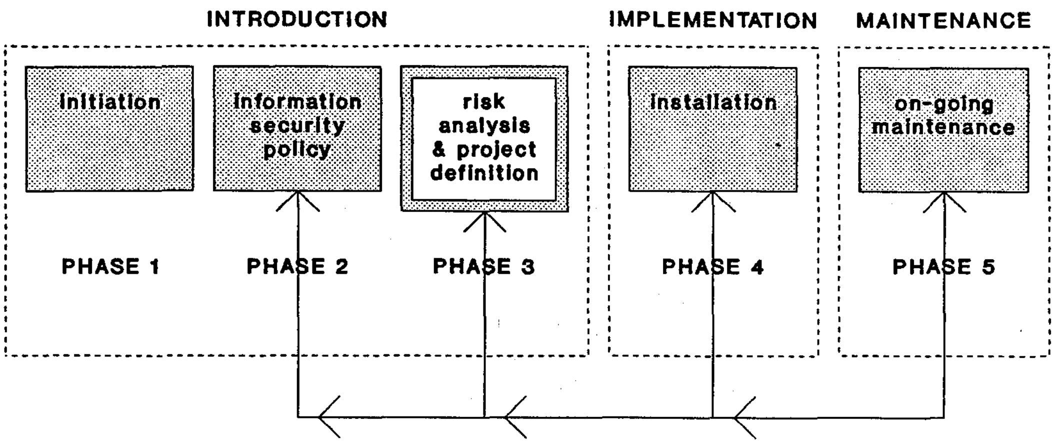


FIG.1 High level view on a methodology for information security

The remainder of this paper therefor aims at addressing approaches to risk analysis as highlighted in phase 3 of the so-called IS-Methodology.

3. THE IRR PHASE

Gathered from a literature overview the following risk related business functions received considerable coverage regarding IRR:

Environmental (especially health related risk)
Engineering (especially nuclear risk)
Finance (especially investment related risk)
Insurance
Computerised Business Information Systems (CBIS)

Many risk related functional philosophies inherent to appropriate **business functions** include risk management in financial terms and plant failure analysis in engineering terms. Within these business functions, various techniques are applied in the process of risk analysis and risk management, such as statistical short term forecasting techniques which are applied in financial risk management.[21] The concept of risk balancing is a technique used in environmental risk resolution. It is clear from the last mentioned that a technique plays an important part in the execution of IRR. The authors therefor decided to import the idea of "**enabling technologies**" (statistical short term forecasting techniques, risk balancing, heuristics, etc.), by so referring to the techniques inherent to specific approaches.

2. THE IS-METHODOLOGY

We believe that another factor that negatively influenced the success rate of IRR applications, is the fact that IRR is often attempted as a standalone exercise and on a piece meal, ad hoc basis. IRR should rather be placed within the context of an overall information security function in an organisation. The position of the IRR phase within an information security methodology can be seen in figure 1. The IS-Methodology as presented usually consists out of five phases:[1]

PHASE 1 - Initiation: The management of an organisation has to be committed to the need for an information security function. Such function should be initiated and guided by a steering committee.

PHASE 2 - Information Security Policy: The definition and acceptance of a formal information security policy, which is in line with organisational strategies and company mission.

PHASE 3 - RISK ANALYSIS AND PROJECT DEFINITION: Information security risks and associated potential losses need to be determined (if possible quantified) and weighed against factors such as productivity, cost of controls, and benefit, in order to select cost-effective safegaurds. The objective of this phase is a well-defined project plan for the installation of the acceptable level of safegaurds.

PHASE 4 - Installation: The timely installation of the information security safegaurds as set out in the project plan.

PHASE 5 - Maintenance: The on-going maintenance includes the review of the status of information security, on a regular basis. It also requires information security program controls to become part of the business analysis and systems development process.

Apart from the concepts **business function** and **enabling technology** one also has to address the issue of the means and utilities that are used with enabling technologies during the risk resolution process. Modelling, monitoring, screening, questionaires and checklists, and computer technology are examples of such means used within business functions, referred to as **risk processes** in the context of our research. The following table illustrates the conceptual relationships between risk related business functions, risk processes and enabling technologies.

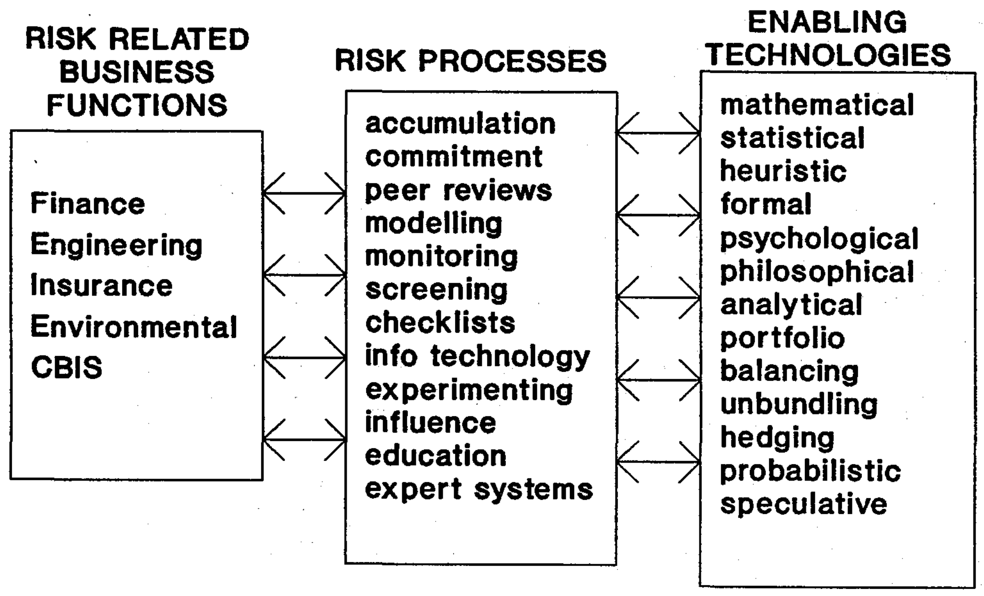


FIG.2 Risk Resolution: The IRR Research Model

From the above diagram can be seen that the IRR Research Approach is made up of the following basic components:

Risk related business functions,
Risk processes, and
Risk resolving enabling technologies.

More than one enabling technology and risk process might be used within each risk related business function, as shown by the literature overview undertaken by the IRR project team. Quantitative methods, for example, which are statistical and mathematical in origin, are applied in the general management process to reduce the risk involved in strategic decision-making. Such a risk resolution exercise can be further facilitated by means of processes such as spreadsheets and computer technology. In the same way more than one business function, enabling technology and risk process might be used within the application of the IRR methodology.

4. RISK RELATED BUSINESS FUNCTIONS

From a literature viewpoint, risks are generally classified into one of two categories, namely (i) risks related to business functions usually associated with socio-psychological issues, and (ii) risks related to technological issues, as indicated by the following diagram:

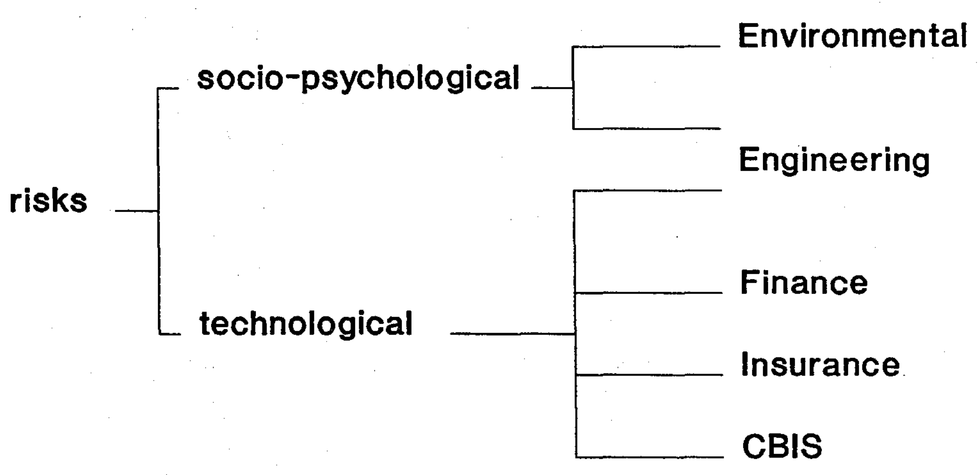


FIG.3 Risk related business functions

The risk related business functions Engineering, Insurance and CBIS will be defined and briefly discussed. Enabling technologies and risk processes within the Environmental and Financial business functions will be discussed in more detail.

4.1 ENGINEERING

From current literature it is clear that risk analysis and risk management in the engineering environment is mostly concerned with system or plant failure analysis. Coverage of nuclear engineering risk assessment constitutes a major part of workshops, seminars, research and subsequent literature. Probabilistic event trees and fault trees are the most prominent enabling technologies applied with regard to risk resolution in the engineering environment. Other approaches are mostly analytical in origin.[23]

4.2 INSURANCE

The theory of risk associated with the insurance industry date as far back as 1909. This classical theory was then mostly associated with life insurance mathematics. The theory of risks has since been expanded to include not only short term insurance risks and other aspects of the insurance business such as reinsurance, but also risks related to strategic decision making in general financial business planning. Enabling techniques applied in insurance risk theory include amongst others, stochastic processes, the time-dependent variation of risk exposure, and the Monte Carlo technique.[4]

4.3 COMPUTERISED BUSINESS INFORMATION SYSTEMS (CBIS)

Software risk management is an emerging discipline whose objectives are to identify, address, and eliminate software risk items. Such risk factors could become either threats to the successful operation of software, or result in major rewrites of software. Enabling technologies and risk processes used in software risk management include amongst others, network analysis, decision trees, risk exposure analysis, the Delphi technique, statistical decision analysis, checklists, cost and performance models.[5]

4.4 ENVIRONMENTAL

Environmental risk is a hazard or danger which threatens the environment, for example the risk of a nuclear accident caused by human error or by natural disaster. It is the probability or chance of an environment (i.e. human, nature, etc.) suffering an adverse consequence, or of encountering some loss. Environmental risk management involves the search for a 'best route' between social benefit and environmental risk. It is a balancing or trading-off process in which various combinations of risks are compared and evaluated against particular social gains.

Risk research has been sponsored by industry and government in many countries, largely because public opposition to some technological development has created powerful contraints on further expansion in, for example, the nuclear power industry.[27,25,24]

4.4.1 ENVIRONMENTAL: THE IMPORTANCE FOR IRR

Risk identification and risk estimation are steps in an environmental risk analysis exercise. Some **risk processes** used within risk identification and estimation include modelling, monitoring (surveillance), testing and screening. **Enabling technologies** include psychological perception, quantitative techniques, heuristics, balancing of risks, probabilistic binary event tree analysis, the concept of reasonableness, and risk rationalisation versus risk reduction.

The following aspects have been addressed in literature on environmental risk assessment. The utilisation of these as enabling technologies in IRR seems interesting and possible:

PSYCHOLOGICAL PERCEPTION

Psychological impact results in social and environmental risk perception to differ greatly from one person to another.[25]

In studies on judgements of positive versus negative values, it has been shown that values guiding our behaviour are more negative on the negative side than positive on the positive side. This means that we are generally more sensitive to increases in loss (i.e. negative risk) than to increases in gains (i.e. positive risk).[25]

In the balance between gains (positive risks) and potential losses (negative risks), or efficiency (positive risk) and risk resolution (controls for negative risk), does the above statement with respect to psychological impact hold true for the IRR environment ? Do information technology managers also regard the risk of the loss of a computer service as having 'more value' than the actual economic benefit of utilising the best information technology to provide a service ?

QUANTITATIVE TECHNIQUES

Difficulty in the quantification of environmental risks is often experienced. There are plenty of examples of risk estimates which are often quoted (e.g. the risk of a disaster at a nuclear power plant), which can very well be uncertain by a factor of 100 or 1000. But as soon as a figure is given, many people tend to forget this and accept the figure as a fact. Quantification of risks in IRR methods have for some time been treated with the same kind of scepticism. This resulted in current research to be aimed at qualitative rather than quantitative approaches.

HEURISTICS

Heuristics have been applied in simplifying environmental risk analysis. This resulted in conclusions being deficient. It is often quite debatable if such conscious deficiency is justified.[25] The same reasoning would apply to IRR. Two thinking paradigms have been identified in the field of IRR, namely rational/analytical versus intuitive/heuristic.[6] It is the author's opinion that the former is obviously more technical whereas the latter is heavily influenced by psychological perception. A general disctintion has been made in literature between risk analysis approaches as being either technological or psychological.[18]

Finance, which has been identified as another major risk related business function, will be discussed next.

4.5 FINANCE

In organisations, risk management in the narrow sense has been dealing with the organisational aspects of assessing and limiting risk. Pure risks are limited to events with detrimental consequences to a company, such as risks threatening assets, labour potential or financial potential of a company and are the result of accidental and probable events. In contradistinction there are speculative risks, which involve the possibility of both gain and loss. The resolution of the latter is usually understood as being financial risk management.[3]

Any financial instrument used within the financial business function, can be viewed as having a unique combination of characteristics, such as yield, duration, size, marketability, and inherent risk profile. Such risk profiles go hand in hand with financial innovation. Financial transactions reallocate various categories of risk among lenders, borrowers and financial intermediaries. The inherent risks associated with finance, include price (market) risk, credit risk, liquidity risk, settlement risk, country and transfer risk, and the investment risk associated with stock trading.[10]

Enabling technologies applied in financial risk management include techniques such as strategic switch analysis, duration and maturity gap analysis, immunisation, portfolio techniques, the unbundling of risks, and quantitative decision tree modelling.

4.5.1 FINANCIAL: THE IMPORTANCE FOR IRR

SWITCH ANALYSIS

Switch analysis is a technique whereby a switch transaction takes place, i.e. the selling of a stock in a portfolio and the simultaneous purchase of a different stock. Owing to different stock volatilities, some stocks will appear to offer better value than others given a particular "view" on interest rates, thus reducing possible negative risk associated with a portfolio of stocks.[10]

Financial switch analysis and the environmental balancing of risks are similar in concept, as they both try to minimise risk to an optimum level. Within the IRR process, instead of reducing risks by means of costly safegaurds, why not use the concept of switching by comparing risks and replacing risks with suitable alternatives ?

MATURITY GAP ANALYSIS

Maturity gap analysis is a flow concept exclusively used for interest rate risk management, while **duration**, as a stock concept, embraces interest rate, investment, and capital risk analysis. Duration and maturity gap analysis may help a bank to fashion financial strategies for the current, or next, financial year that will give it the accounting profits it needs.

There also seem to be some similarity between the time-dependent variation of risk exposure used in insurance risk theory and the duration and maturity gap analysis techniques used in financial risk management, as both involve time factors. The time-change factor also plays an important role in IRR, because of the dynamic character of the information technology environment.

PORTFOLIO THEORY

The central idea of **portfolio theory** is that the total risk of an investment can be reduced by spreading it over a pool of assets.[3]

In the application of financial portfolio techniques, the application of a risk reducing measure is comparable to an investment in an asset. If the security of certain values is based on a single measure, the total of values at risk is exposed if the measure fails. If, however, a combination of measures, viz. a portfolio of measures, has been applied, the failure of an individual component will still result in a recuded risk.[3]

5. CONCLUSION

In this paper the question "Which approach combined with underlying business philosophies and business technologies ?" instead of "Which package ?" has been addressed, because the authors felt that research into underlying business philosophies related to risk analysis could contribute in resolving the dilemma that so often governs the application of IRR. The authors also strongly support the concept that IRR should be placed within the context of an overall information security methodology, such as the IS-Methodology.

The basic components of the IRR research model have been identified as: Business Functions, Risk Processes and Enabling Technologies. The business functions Environmental and Finance have been discussed so to demonstrate the applicability of these concepts to the issues surrounding Information Risk Resolution. The discussion on Environmental risk analysis appears to be very appropriate to the much discussed topic of Disaster Recovery Planning for the computer facilities of an organisation. The possibility of applying some of these enabling technologies in IRR raises the question: how can they be adapted for the information technology environment ? The last mentioned requires further research and will be reported on in a follow-up paper.

REFERENCES

[1] Badenhorst K.P. & Eloff Jan H.P., "Framework of a Methodology for the Life Cycle of Computer Security in an Organisation", Computers & Security, 8 (1989) 433-442, Elsevier Science Publishers Ltd.

[2] Badenhorst K.P. & Eloff Jan H.P., "Computer Security Methodology: Risk Analysis and Project Definition", Computers & Security, 9 (1990) 339-346, Elsevier Science Publishers Ltd.

[3] Bauknecht Kurt & Strauss Christine, "Portfolio techniques to Support Risk Management and Security", IFIP/Sec 90 on 'Computer Security and Integrity in our Changing World', The 6th International Conference and Exhibition on Information Security, Espoo (Helsinki), Finland, May 23-25 1990.

[4] Beard R.E., Pentikäinen T. & Pesonen E., "Risk Theory, The Stochastic Basis of Insurance", Third Edition, Chapman and Hall, London, 1984.

[5] Boehm Barry W., "Software Risk Management", IEEE Computer Society Press, Washington D.C., 1989.

[6] Caelli William J., "Information Security: The Next Decade", Seminar organised by Computer Society of South Africa in conjunction with IFIP Committee (TC11), Cape Town, South Africa, 18 May 1990.

[7] Carroll John M., "Coding Ethics and Law into Risk Analysis", Proceedings of Compsec 90 International, London, 10-12 October 1990.

[8] Crouch Edmund A.C. & Wilson Richard, "Risk/Benefit Analysis", Ballinger Publishing Company, Cambridge Massachusetts, 1982.

[9] Dorey Dr Paul G, "Computer Security Risk Management: Practical Experiences of a User", Proceedings of Compsec 90 International, London, 10-12 October 1990.

[10] Falkena H.B. & Kok W.J. (Ed.), "Essays on Financial Risk Management", Macmillan Press Ltd, London, 1988.

[11] Gardner Philip E., "Evaluation of Five Risk Assessment Programs", Computers & Security, 8 (1989) 479-485, Elsevier Science Publishers.

[12] Gilbert Irene E., "Automated Risk Management Software Tools", National Computer Systems Laboratory & Computer Security Management and Information Exchange Group, National Institute of Standards and Technology, Gaithersburg, Maryland, U.S.A., 1990.

[13] Godfrey A.I., "Quantitative Methods for Managers", Edward Arnold Publishers Ltd, London, 1977.

[14] Guarro Sergio B., "Principles and Procedures of the LRAM Approach to Information Sysyems Risk Analysis and Management", Computers & Security, 6 (1987) 493-504, Elsevier Science Publishers.

[15] Hutt A.E., Bosworth S. & Hoyt D.B., "Computer Security Handbook", Second Edition, Macmillan Publishing Company, New York, 1988, pp.22-32, 299.

[16] Katzke Dr. Stuart W., "NBS Perspectives on Risk Analysis: Past, Present and Future", from Minutes of the Federal Informaiton Systems Risk Analysis Workshop, The Air Force Computer Security Program Office, U.S.A., 1985, pages 2.3-2.5.

[17] Kunreuther Howard (Ed.), "Risk: A Seminar Series", IIASA Collaborative Proceedings Series, CP-82-S2, International Institute for Applied Systems Analysis, Laxenburg Austria, 1982.

[18] Kunreuther Howard C. & Ley Eryl V (Ed.), "The Risk Analysis Controversy: An Institutional Perspective", Proceedings of a Summer Study on Decision Processes and Institutional Aspects of Risk held at IIASA, Laxenburg, Austria, 22-26 June 1981, Springer Verlag Berlin 1982.

[19] Lamère J.-M., Leroux Y. & Tourly J., "La Sécurité des Réseaux, Méthodes at Techniques", Dunod Informatique, Bordas, Paris 1987.

[20] Moses Robin H & Glover Ian, "A Model of Risk Analysis and Management", Proceedings of the 2nd Annual Canadian National Computer Security Conference, Ottawa, March 1990.

[21] O'Donovan T.M., "Short Term Forecasting: An Introduction to the Box-Jenkins Approach" John Wiley & Sons Ltd., Chichester, 1983.

[22] Proceedings: IFIP/Sec 90 (Espoo Helsinki Finland) 23-25 May 1990, Campsec 90 (London U.K.) 10-12 October 1990, 13th National Computer Security Conference (Washington D.C. U.S.A.) 1-4 October 1990.

[23] Proceedings of the International Ans/Ens Topical Meeting on PROBALISTIC RISK ASSESSMENT - September 20-24, 1981, Port Chester, New York. Sponsored by AMERICAN NUCLEAR SOCIETY, EUROPEAN NUCLEAR SOCIETY.

[24] Shrader-Frechette K.S., "Risk Analysis and Scientific Method", D.Reidel Publishing Company, Dordrecht 1985.

[25] Sjöberg Lennart (Ed.), "Risk and Society", The Risks and Hazards Series: 3, Allen & Unwin (Publishers) Ltd., London, 1987, p 156.

[26] Smith S.T. & Lim J.J., "Framework for Generating Expert Systems to Perform Computer Security Risk Analysis", First Annual Armed Forces Communications and Electronics Association Symposium and Exposition on Physical and Electronics Security, Philadelphia, August 19-21, 1985.

[27] Whyte Anne V. & Burton Ian (Ed.), "Environmental Risk Assessment", Scope 15, John Wiley and Sons, Toronto, 1980.

INFORMATION SYSTEMS SECURITY: A COMPREHENSIVE MODEL

Capt John R. McCumber
Joint Staff/J6K
The Pentagon
Washington, DC 20318-6000

INTRODUCTION

At speech to the 13th National Computer Security Conference on 3 October 1990, Michelle VanCleave, Assistant Director for National Security Affairs, Executive Office of the President stated, "We need a comprehensive model for understanding the threat to our automated information systems." I believe I have developed that model. This model not only addresses the threat, it functions as an assessment, systems development, and evaluation tool. The model is unique in that it stands independent of technology. Its application is universal and is not constrained by organizational differences. As with all well-defined fundamental concepts, it is unnecessary to alter the premise even as technology and human understanding evolve.

Computers communicate. Communication systems compute. The evolution of technology has long since eliminated any arbitrary distinction between a computer and its communication components or a communications network and its computing system. Some organizations have attempted to deal with the phenomenon by marrying these functions under common leadership. This has resulted in hyphenated job descriptions such as Computer-Communications Systems Staff Officer and names like Information Technology Group. Unfortunately, these names can mask an inappropriate or poorly executed realignment of organizational responsibilities. Ideally, management will recognize there is a theoretical-as well as organizational-impact.

The same is true for the security disciplines. Merely combining the communications security (COMSEC) and computer security (COMPUSEC) disciplines under an umbrella of common management is unacceptable. Even if we address the other, albeit less technical, aspects of information systems security such as policy, administration, and personnel security, we still fail to develop a comprehensive view of this evolving technology. The reason for this becomes clear when we are reminded it's the information that is the cornerstone of information systems security. In this sense, any paradigm which emphasizes the technology at the expense of information will be lacking.

THE NATURE OF INFORMATION

Defining the nature of information could be a tedious task. To some it represents the free-flowing evolution of knowledge; to others, it is intelligence to be guarded. Add to this the innumerable media through which information is perceived and we have a confusing array of contradictions. How can we present a study of information that has universal application?

It may be best to develop a simple analogy. The chemical compound H_2O means many things to all of us. In its liquid state, water means life-giving sustenance to a desert-dwelling Bedouin; to a drowning victim, it is the vehicle of death. The same steam we use to prepare vegetables can scald an unwary cook. Ice can impede river-borne commerce on the Mississippi River or make a drink more palatable. Science, therefore, does not deal with the perception of the compound, but with its state.

As the compound H_2O can be water, ice, or steam, information has three basic states which I've already depicted. At any given moment, information is being transmitted, stored, or processed. The three states exist irrespective of the media in which information resides. This subtle distinction ultimately allows us to encompass all information systems technology in our model.

It is possible to look at the three states in microcosm and say that processing is simply specialized state combinations of storage and transfer; so, in fact, there are only two possible states. By delving to this level of abstraction, however, we go beyond the scope and purpose of the model. The distinction between the three states is fundamental and necessary to accurately apply the model. For example, cryptography can be used to protect information while it's transferred through a computer network and even while it is stored in magnetic media. However, the information must be available in plaintext (at least to the processor) in order for the computer to perform the processing function. The processing function is a fundamental state which requires specific security controls.

When this information is needed to make a decision, the end user may not be aware of the number of state changes effected. The primary concern will be certain characteristics of the information. These characteristics are intrinsic and define the security-relevant qualities of the information. As such, they are the next major building block of our information systems security model.

CRITICAL INFORMATION CHARACTERISTICS

Information systems security concerns itself with the maintenance of three critical characteristics of information: confidentiality (Pfleeger's "secrecy"), integrity, and availability [PFL89]. These attributes of information represent the full spectrum of security concerns in an automated environment. They are applicable for any organization irrespective of its philosophical outlook on sharing information.

CONFIDENTIALITY

Confidentiality is the heart of any security policy for an information system. A security policy is the set of rules that, given identified subjects and objects, determines whether a given subject can gain access to a specific object [DOD85]. In the case of discretionary access controls, selected users (or groups) are controlled as to which data they may access. Confidentiality is then the assurance that access controls are enforced. The reason

I prefer the term confidentiality to secrecy is merely to avoid unwarranted implications that this is solely the domain of armies and governments. As we will see, it is a desirable attribute for information in any organization.

All organizations have a requirement to protect certain information. Even owners of a clearinghouse operation or electronic bulletin need the ability to prevent unwanted access to supervisory functions within their system. It's also important to note the definition of data which must be protected with confidentiality controls is broadening throughout government [OTA87]. Actual information labeling and need-to-know imperatives are aspects of the system security policy which are enforced to meet confidentiality objectives. The issue of military versus civilian security controls is one which need not impact the development of a comprehensive representation of information systems security principles.

INTEGRITY

Integrity is perhaps the most complex and misunderstood characteristic of information. As I stated, we seem to have a better foundation in the development of confidentiality controls than those which can help insure data integrity. Pfleeger defines integrity as "assets (which) can only be modified by authorized parties" [PFL89]. Such a definition unnecessarily confines the concept to one of access control.

I propose a much broader definition. Data integrity is a matter of degree (as is the concept of "trust" as applied to trusted systems) which has to be defined as a quality of the information and not as who does/does not have access to it. Integrity is that quality of information which identifies how closely the data represent reality. How closely does your resume reflect "you"? Does a credit report accurately reflect the individual's historical record of financial transactions? The definition of integrity must include the broad scope of accuracy, relevancy, and completeness.

Data integrity calls for a comprehensive set of aids to promote accuracy and completeness as well as security. This is not to say that too much information can't be a problem. Data redundancy and unnecessary records present a variety of challenges to system implementors and administrators. The users must define their needs in terms of the information necessary to perform certain functions. Information systems security functions help insure this information is robust and (to the degree necessary) reflects the reality it is meant to represent.

AVAILABILITY

Availability is a coequal characteristic with confidentiality and integrity. This vital aspect of security insures the information is provided to authorized users when it's requested or needed. Often it's viewed as a less technical requirement which is satisfied by redundancies within the information system such as back-up power, spare data channels, and parallel data bases. This perception, however, ignores one of the most valuable aspects of our model which this characteristic provides. Availability is the check-and-balance constraint on our model. Because security and utility often conflict, the science of information systems security is also a study of subtle compromises.

As well as insuring system reliability, availability acts as a metric for determining the extent of information system security breaches [DOJ88]. Ultimately, when information systems security preventive measures fail, remedial action may be necessary. This remedial activity normally involves support form law enforcement or legal departments. In order to pursue formal action against people who abuse information systems resources, the ability to prove an adverse impact often hinges on the issue of denying someone the availability of information resources. Although violations of information confidentiality and integrity can be potentially more disastrous, denial of service criteria tend to be easier to quantify and thus create a tangible foundation for taking action against violators [CHR90].

The triad of critical information characteristics covers all aspects of security-relevant activity within the information system. By building a matrix with the information states positioned along the horizontal axis and the critical information characteristics aligned down the vertical, we have the foundation for the model.

SECURITY MEASURES

We've now outlined a matrix which provides us with the theoretical basis for our model. What it lacks at this stage is a view of the measures we employ to insure the critical information characteristics are maintained while information resides in or moves between states. It's possible, at this point, to perceive the chart as a checklist. At a very high level of abstraction, one could assess the security posture of a system by using this approach. By viewing the interstices of the matrix as a system vulnerability, you can attempt to determine the security aspects of an information system as categorized by the nine intersection areas. For example, you may single out systems information confidentiality during transmission or any intersection area for scrutiny.

The two-dimensional matrix also has another less obvious utility. We can map various security technologies into the nine interstices. Using our example from above, we note it is necessary to protect the confidentiality of the information during its transmission state. We can then determine which security technologies help insure confidentiality during transmission of the information. In this case, cryptography would be considered a primary security technology. We can then place various cryptographic techniques and products within a subset in this category. Then we repeat the process with other major types of technology which can be placed within this interstice. The procedure is repeated for all nine blocks on our grid. Thus we form the first of three layers which will become the third dimension of our model-security measures.

TECHNOLOGY

The technology layer will be the primary focus of the third dimension. We will see that it provides the basis for the other two layers. For our purposes, we can define technology as any physical device or technique implemented in physical form which is specifically used to help insure the critical information characteristics are maintained through any of the information states. Technology can be implemented in hardware, firmware, or software. It could be a biometric device, cryptographic module, or security-enhanced operating system. When we think of a thing which could be used to protect the critical characteristics of information, we are thinking of technology.

Usually, organizations are built around functional responsibilities. The advent of computer technology created the perception that a group needed to be established to accommodate the new machines which would process, store, and transmit much of our vital information. In other words, the organization was adapted to suit the evolving technology. Is this wrong? Not necessarily; however, it is possible to create the impression that technology exists for technology's sake. Telecommunications and computer systems are simply media for information. The media need to be adapted to preserve certain critical characteristics with the adaptation and use of the information media (technology). Adaptation is a design problem, but use and application concerns bring us to the next layer.

POLICY AND PRACTICE

The second layer of the third dimension is that of policy and practice. It's the recognition of the fact that information systems security is not just a product which will be available at some future date. Because of our technology focus, it's easy to begin to think of security solutions as devices or add-on packages for existing information systems. We are guilty of waiting for technology to solve that which is not solely a technological problem. Having an enforceable (and enforced) policy can aid immeasurably in protecting information.

A study has shown 75% of Federal agencies don't have a policy for the protection of information on PC-based information systems [OTA87]. Why, if it is so effective, is policy such a neglected security measure? It may be due in part to the evolving social and moral ethic with regard to our use of information systems. The proliferation of unauthorized software duplication is just another symptom of this problem. Even though software companies have policies and licensing caveats on their products, sanctions and remedies allowed by law are difficult if not impossible to enforce. No major lawsuit involving an individual violator has come before our courts, and it appears many people don't see the harm or loss involved. Although there are limits established by law, it seems we as "society" accept a less stringent standard.

Closely associated with the matter of policy is that of practice. A practice is a procedure we employ to enhance our security posture. For example, we may have a policy which states that passwords must be kept confidential and may only be used by the uniquely-authenticated user. A practice which helps insure this policy is followed would be committing the password to memory rather than writing it somewhere.

The first two layers of the third dimension represent the design and application of a security-enhanced information system. The last building block of our model represents the understanding necessary to protect information. Although an integral aspect of the preceding two layers, it must be considered individually as it is capable of standing alone as a significant security measure.

EDUCATION, TRAINING, AND AWARENESS

The final layer of our third dimension is that of education, training, and awareness. As you will see, were the model laid on its back like a box, the whole model would rest on this layer. This phenomenon is intentional. Education, training and awareness may be our most prominent security measures, for only by understanding the threats and vulnerabilities associated with our proliferating use of automated information systems can we begin to attempt to deal effectively with other control measures.

Technology and policy must rely heavily on education, training, and awareness from numerous perspectives. Our upcoming engineers and scientists must understand the principles of information security if we expect them to consider the protection of information in the systems they design. Currently, nearly all university graduates in computer science have no formal introduction to information security as part of their education [HIG89].

Those who are responsible for promulgating policy and regulatory guidance must place bounds on the dissemination of information. They must insure information resources are distributed selectively and securely. The issue is ultimately one of awareness. Ultimate responsibility for its protection rests with those individuals and groups which create and use this information; those who use it to make critical decisions must rely on its confidentiality, integrity, and availability. Education, training, and awareness promises to be the most effective security measure in the near term.

Which information requires protection is often debated in government circles. One historic problem is the clash of society's right to know and an individual's right to privacy. It's important to realize that these are not bipolar concepts. There is a long continuum which runs between the beliefs that information is a free flowing exchange of knowledge and that it is intelligence which must be kept secret. From a governmental or business perspective, it must be assumed that <u>all</u> information is intelligence. The question is not should information be protected, but how do we intend to protect the confidentiality, integrity, and availability of it within legal and moral constraints?

THE MODEL

OVERVIEW

The completed model appears as Figure 1. There are nine distinct interstices, each three layers deep. All aspects of information systems security can be viewed within the framework of the model. For example, we may cite a cryptographic module as technology which protects information in its transmission state. What many information system developers fail to appreciate is that for every technology control there is a policy (sometimes referred to as doctrine) which dictates the constraints on the application of that technology. It may also specify parameters which delimit the control's use and may even cite degrees of effectiveness for different applications. Doctrine (policy) is an integral yet distinct aspect of the technology. The third layer-education, training, and awareness-then functions as the catalyst for proper application and use of the technology based on the policy (practice) application.

Not every security measure begins with a specific technology. A simple policy or practice often goes a long way in the protection of information assets. This policy or practice is then effected by communicating it to employees through the education, training, and awareness level alone. This last layer is ultimately involved in all aspects of the information systems security model. It may also be solely an educational, training, or awareness security control. The model helps us understand the comprehensive nature of information security that a COMSEC/COMPUSEC perspective cannot define.

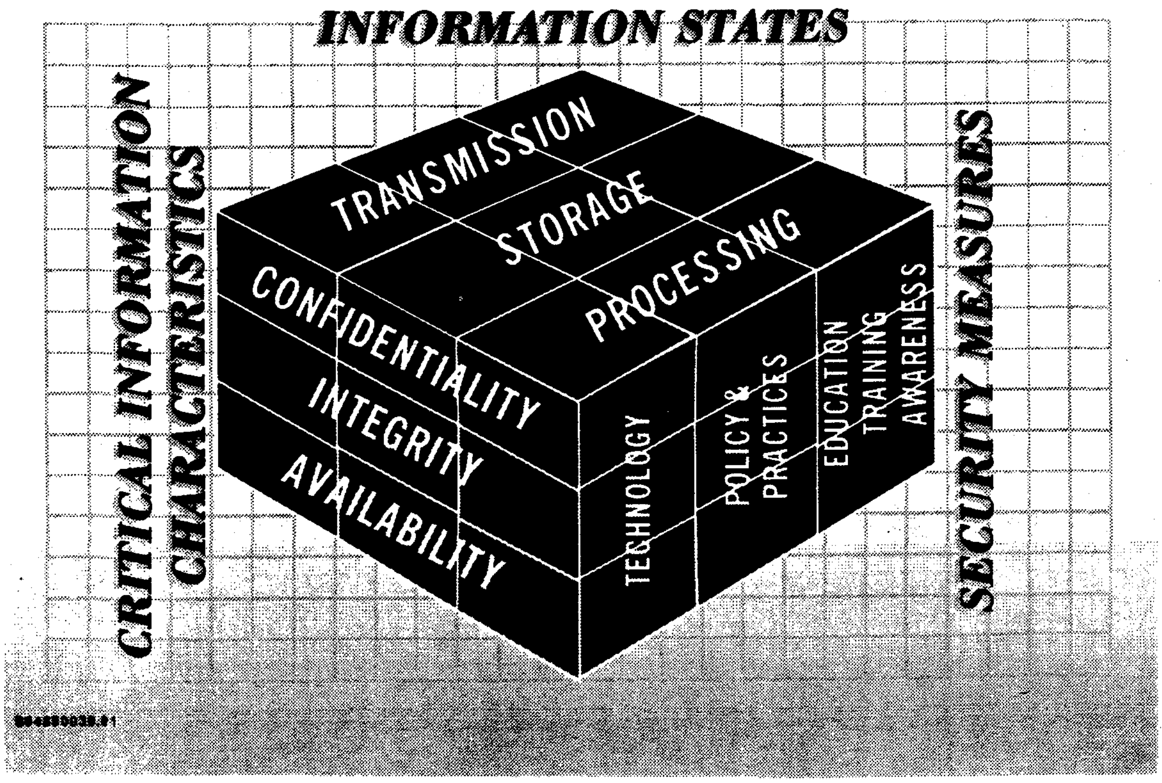


Figure 1

USE OF THE MODEL

The model has several significant applications. Initially, the two-dimensional matrix is used to identify information states and system vulnerabilities. Then, the three layers of security measures can be employed to minimize these vulnerabilities based on a knowledge of the threat to the information asset. Let's take a brief look at these applications.

A developer would begin using the model by defining the various information states within the system. When an information state is identified, one then works down the vertical path to address all three critical information characteristics. Once vulnerabilities are noted in this fashion, it becomes a simple matter of working down through the three layers of security measures. If a specific technology is available, the designer knows that policy and practice as well as education, training, and awareness will be logical follow-on aspects of that control. If a technology cannot be identified, then policy/practice must be viewed as the next likely avenue. (Again, the last layer will be used to support the policy/practice.) If none of the first two layers can satisfactorily counter the vulnerability then, as a minimum, an awareness of the weakness becomes important and fulfills the dictates of the model at the third layer.

Another important application is realized when the model is used as an evaluation tool. As in the design and development application, the evaluator first identifies the different information states within the system. These states can be identified separately from any specific technology. A valuable aspect of the model is the designer needn't consider the medium.

After identifying all the states, an evaluator or auditor can perform a comprehensive review much the same way the systems designer used the model during the development phase. For each vulnerability discovered, the same model is used to determine appropriate security measures. The third dimension of the model insures the security measures are considered in their fullest sense. It is important to note that a vulnerability may be left unsecured (at an awareness level in the third layer) if the designer or evaluator determines no threat to that vulnerability exists. Although no security practitioner should be satisfied with glaring vulnerabilities, a careful study of potential threats to the information may disclose that the cost of the security measure is more than the loss should the vulnerability be exploited. This is one of the subtle compromises alluded to earlier.

The model can also be used to develop comprehensive information systems security policy and guidance necessary for any organization. With an accurate understanding of the relation of policy to technology and education, training, and awareness, you can insure your regulations address the entire spectrum of information security. It's of particular importance that corporate and government regulations not be bound by technology. Use of this model allows management to structure its policy outside the technology arena.

The model functions well in determining requirements for education, training, and awareness. Since this is the last layer, it plays a vital role in the application of <u>all</u> the security measures. Even if a designer, evaluator, or user determines to ignore a vulnerability (perhaps because of a lack of threat), then the simple acknowledgement of this vulnerability resides in the last layer as "awareness". Ultimately, all technology, policies, and practices must be translated to the appropriate audience through education, training, and awareness. This translation is the vehicle which makes all security measures effective. For a more complete understanding of the nuances of education, training, and awareness see [MAC89].

The twenty-seven individual "cubes" created by the model can be extracted and examined individually. This key aspect can be useful in categorizing and analyzing countermeasures. It's also a tool for defining organizational responsibility for information security. The example shows a policy security measure for protecting the confidentiality of information while it is being processed. By considering all 27 such "cubes", the analyst is assured of a complete perspective of all available security measures. Unlike other computer security standards and criteria, this model connotes a true "systems" viewpoint.

CONCLUSION

The information systems security model acknowledges information, not technology, as the basis for our security efforts. The actual medium is transparent in the model. This eliminates unnecessary distinctions between COMSEC, COMPUSEC, TECHSEC, and other technology-defined security sciences. As a result, we can model the security relevant processes of information throughout an entire information system-automated or not. This important aspect of the model eliminates significant gaps in currently-used security architecture guidance for information systems.

I developed this model to respond to the need for a theoretical foundation for modeling the information systems security sciences. The organizational realignments which have recognized the interdependence of several complementary technologies will need refinement in the near future. We can begin that process now by acknowledging the central element in all our efforts-information. Only when we build on this foundation will we accurately address the needs of information systems security in the next decade and beyond.

REFERENCES

[CHR90] Interview with Agent Jim Christy, Chief, Air Force Office of Special Investigations, Computer Crime Division, 26 March 1990

[DOD85] Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, Department of Defense, Washington, DC, December 1985

[DOJ88] Basic Considerations in Investigating and Proving Computer-Related Federal Crimes, U.S. Department of Justice, Justice Management Division, Washington, DC, November 1988

[HIG89] Higgins, John C., Information Security as a Topic in Undergraduate Education of Computer Scientists, Proceedings of the 12th National Computer Security Conference, November 1989

[MAC89] Maconachy, W.V., Computer Security Education, Training, and Awareness: Turning a Philosophical Orientation into Practical Reality, Proceedings of the 12th National Computer Security Conference, November 1989

[OTA87] U.S. Congress, Office of Technology Assessment, Defending Secrets, Sharing Data: New Locks and Keys for Electronic Information, OTA-CIT-310, Washington, DC: U.S. Government Printing Office, October 1987

[PFL89] Pfleeger, Charles P., Security in Computing, Prentice-Hall, 1989

INTEGRATING B2 SECURITY INTO A UNIX SYSTEM

Kevin Brady

UNIX System Laboratories, Inc.
190 River Road, Summit NJ 07901

Overview

Within the last few years the integrity of many computer systems has been violated in a variety of ways, the most prevalent of which has been via "virus" attacks. These attacks feature software which, either intentionally or accidentally, result in a compromise of system security and subsequently result in hundreds of thousands of dollars of damage in the form of compromised/lost data or computer downtime. Currently, most attacks are detected long after the fact. Unfortunately, by the time the intrusion is detected, significant damage is done. In the case of a virus, it is likely to have spread throughout an entire network of computers.

With the advent of systems containing additional security features such as access control lists, least privilege, and mandatory access control, the question arises, do these systems meet the challenge of preventing system security violations and containing virus programs while still retaining the "look and feel" of a traditional UNIX system ?

This paper focuses on the features added to UNIX System V Release 4.1 Enhanced Security (SVR4.1ES) intended to raise the overall level of system security to the B2/F-B2 level.

1. Motivation

By the late 1980's, increased concerns regarding the privacy of computerized data, fear of unauthorized access, and concerns regarding system and data integrity led to a demand within the UNIX community for a higher level of system security. This in turn led to the inclusion of enhanced security features such as those present in SVR4.1ES. While the model for some enhanced features, such as mandatory access control (MAC), have their origins with the Trusted Computer Systems Evaluation Criteria (TCSEC), many others, such as discretionary access control, represent extensions of existing features within the UNIX system. The combination of these features, specifically least privilege and enhanced access control (MAC & DAC), not only provides an environment that is more resistant to penetration and compromise than the UNIX systems that preceded it but also provides compatibility for existing applications and retains the "look and feel" of the UNIX system.

The following is a brief discussion of the approach used for feature definition followed by a description of the key features found in the SVR4.1ES system; Least Privilege/Trusted Facility Management, Enhanced Access Control (Mandatory & Discretionary), Trusted Path, and Auditing.

2. Least Privilege

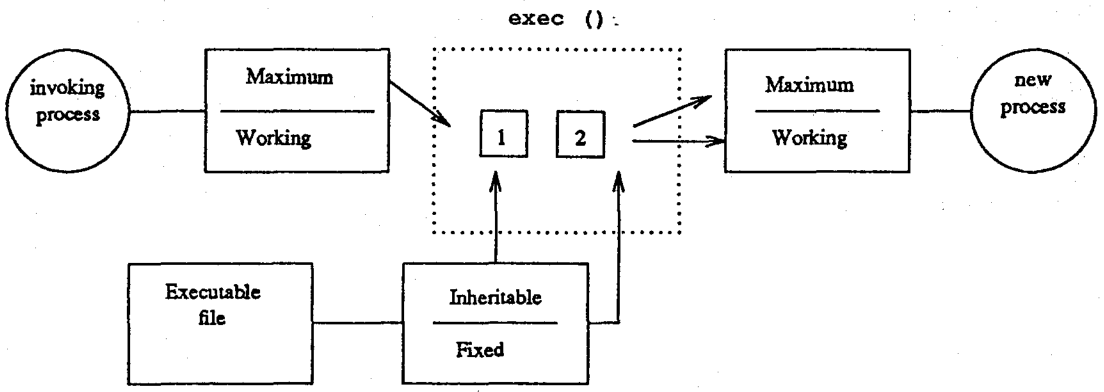
A frequent form of system security subversion is accomplished by the acquisition of "super user" or UID 0 privileges. Historically the UNIX system had a single privileged identity, that of "root" assigned the User Id (UID) of 0. Both file access rights and privilege (i.e., the ability to circumvent the system security policy) were based upon the UID. Due to the dual nature of the UID, once the all powerful user identity of "root" was acquired, the attacker was then able to freely circumvent the system security policy, usually without detection. This type of attack exploits several weaknesses with the historical "root"/UID 0 permission/privilege scheme.

The SVR4.1ES least privilege feature provides the ability for administrators to invoke tasks requiring privilege without requiring "root" access. In previous versions of UNIX, any attempt to execute a sensitive system service (e.g., mount a file system) required the use of a "privilege." In System V, there has been traditionally one such privilege, commonly called "root" or "superuser", which is signified by a

UNIX is a registered trademark of UNIX System Laboratories, Inc.

process whose effective user id is 0. In SVR4.1ES, this single superuser privilege is subdivided into a finer grain set of privileges designed to ensure that sensitive system services execute with the minimum amount of privilege required to execute the task.

In SVR4.1ES, a process has a **maximum** and **working** set of privileges associated with it. The **maximum** set represents the most privilege the process could ever attain and the **working** set represents the minimum set of privileges required to execute the task. A executable file may have associated with it an **inheritable** or **fixed** set of privileges. A **inheritable** privilege is a privilege which is kept (i.e., left "turned on") only if it already existed in the process. A **fixed** privilege is a privilege which is always given to the process independent of the previous process privileges. When a file is exec'ed these sets are computed as illustrated in the following diagram:



(1) Intersection Of Maximum Set Of Privileges Of The Invoking Process With The Inheritable Privileges Of The File

(2) Union Of The Results Of (1) With The Fixed Privileges Of The File

Note: The **fixed** and **inheritable** privilege sets are disjoint; a privilege cannot be present in both sets at the same time.

For compatibility with the current UNIX **setuid** mechanism, SVR4.1ES supports the concept of **fixed** file privileges. When a file is executed that has fixed privilege(s), those privilege(s) are added (unioned) with the **maximum** privilege set of the invoking process forming the **maximum** and **working** privilege sets for the resulting process. Note that the **fixed** privileges are not added to the **maximum** or **working** privilege sets of the invoking process.

For example if a site determined that all users should be able to execute the *ps* command and not be subject to mandatory or discretionary access control checks, the administrator would use the ***filepriv*** command to set the **p_DACread** and **p_MACread** privileges as **fixed** privileges. Any user invoking *ps* would then acquire the **p_DACread** and **p_MACread** privileges for the duration of the execution of the *ps* command.

For an additional degree of protection, system applications are written such that all privileges in the **working** set are turned off prior to ***exec***. Thus the exec'ed process must explicitly set the privileges which it requires to properly execute. Since all privileges in the **working** set are dropped prior to ***exec***, even if a rogue version of a command were executed it would have inherited no privileges, thus no damage would have occurred. Note that only the active privileges (i.e., the **working** set) were dropped. This allows a properly exec'ed application to turn on the correct set of privileges upon execution (since the privileges still exist in the **maximum** set).

2.0.1 Trusted Facility Administration (TFM)

The trusted facility administration (***tfadmin***) facility redefines the way in which the role/privilege assignment mechanism works. In current UNIX systems, an administrator will ***login*** (or ***su***) to an administrative identity. Upon assumption of the identity, all file access rights (and privileges in the case of "root"/UID 0) associated with the identity are assumed by the administrator; all subsequent processes assume these privileges. With this in mind, there are several scenarios by which the vulnerabilities of the system may be exploited. For example logged in as "root" the administrator invokes:

```
$ date 010191 (set system date & time)
$ mail
```

Since a full pathname was not specified the administrator is relying on the PATH variable being properly set such that the correct commands are executed. Thus the administrator is very vulnerable to attack via trojan horse programs. In this example if the administrator's PATH is not properly set (likely if the administrator assumed the identity via *su*), rogue versions of ***mail*** or ***date*** could be executed resulting in the administrator giving "root" privileges away unknowingly. Since all of the attributes associated with the "root" identity are passed to child processes via ***exec***, all processes invoked by the administrator execute with privilege, regardless of need. This in turn often results in the execution of code which is not expecting to run with "root" privilege and was not designed with trust in mind. This is especially dangerous with commands that in turn execute other commands or that feature escapes to the shell. For example, an administrator escapes to the shell from ***mail*** and executes ***cat***. Since ***mail*** was running as "root", the ***cat*** command was also executed as "root". If a rogue version of ***cat*** was executed, "root" privilege has inadvertently been given away.

With ***tfadmin*** there are no privileges inherent with a given user identity, rather privileges are associated with a defined role and are only acquired through execution of ***tfadmin***. The ***tfadmin*** command has associated with it an administrator controlled data base. The data base contains entries in the following format:

role:alias:command:privilege(s)

-for example-

secadmin:date:/bin/date:p_sysops

Considering the example above:

```
$ tfadmin date 010191
$ mail
```

Upon execution, the ***tfadmin*** command searches its database for an entry for ***date*** for the "role" invoking ***tfadmin***. If a match is found, the command is executed (via its fully qualified pathname) only with the explicit privileges needed to perform the requested operation. In this case, only the sysops privilege is needed to set the date, thus this is the only privilege passed to the process executing ***date***. The next command ***mail*** requires no privilege to run, therefore execution via ***tfadmin*** is unnecessary. Since ***tfadmin*** will only associate privilege with a defined entry, if the administrator invoked:

tfadmin mail

the command would fail since no database entry would be defined for ***mail*** (since ***mail*** does not require privileged execution).

3. Mandatory Access Control

In order to meet customer needs for high data integrity, Mandatory Access Control (MAC) labels have been added to SVR4.1ES. With the addition of Mandatory Access Control, all processes, files, and IPC objects must have a security label. While the DAC mechanism allows permissions to be set at the discretion of the owner of an object, the MAC mechanism is set by the system administrator and enforced by the system. The mandatory access control policy follows a modified Bell-LaPadula model [2] that can be summarized as "read equal or down" and "write equal." For instance, a process at level "top-secret" can read a file at level "secret," and a process at level "secret" would only be able to write to a file at level "secret."

Administrators are responsible for determining and setting up the discrete set of labels at which a user can log in. An administrator also sets a login level range on a terminal line, such that when a user attempts to login, the label specified by the user must dominate the login-low label on the terminal line and in turn be dominated by the login-high label on the terminal line.

By default, SVR4.1ES supports 256 classifications and 1024 categories though the system can be configured to support values up to 65535 and 2097152. For reasons of disk space and performance, SVR4.1ES implements MAC labels with an "indirection" scheme. Each named classification/category tuple (i.e., fully qualified label) is associated with a unique level identifier also known as a LID. The LID serves as a system "pointer" to the fully qualified label name and is the value which is stored in the inode. For reasons of user convenience, each fully qualified label may be assigned an "alias" name. The "alias" name is a short hand representation of the fully qualified label. For example, the "alias" for the label:

TopSecret:projectA,projectB

may be: **TS**

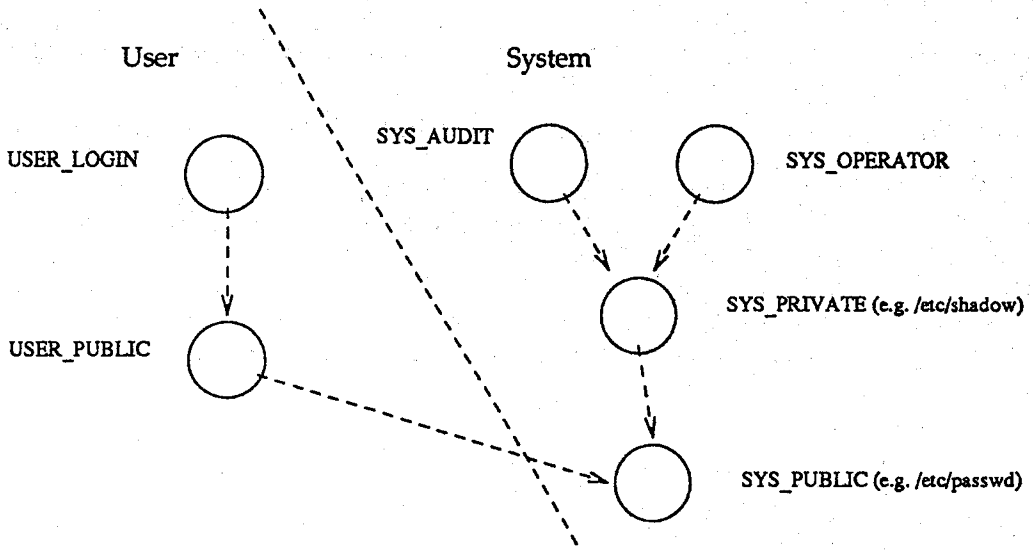
The kernel uses the **LID** as the primary method of label reference. When the kernel is requested to check access, the LIDs involved in the access determination are compared. If **write** access is requested, the LIDs themselves are simply compared (since the system enforces a policy of write equal and the LIDs are guaranteed to be unique). For example, if **write** access to a file with a lid of **10045** is requested by a process with a LID of **10045**, access is granted since the LIDs are equal. However if **write** access is requested to the same file by a process with a LID of **10046** access is denied since the LIDs are not equal. Since the system supports a policy of "read down" the access check required for a **read** operation requires an additional step. Since no hierarchy can be determined by the comparison of two LIDs (i.e., LID **10046** is not guaranteed to dominate LID **10045**), the **binary representation** of the fully qualified labels of the two LIDs needs to be compared. For reasons of system performance, the **binary representation** of the labels are kept in a cache, the size of which is a system tunable that may be increased or decreased as required. For example if a **read** operation was requested to a file with a LID of **10045** by a process with a LID of **10046**, the system would do the following:

- Check to see if the binary representation of the LIDs to be compared is already in the cache.
- If the binary representation of both LIDs are not in the cache, the system reads the LID database and brings the binary representation of the LID(s) into the cache.
- The binary representation of the LIDs are compared to determine if a **dominance** relationship exists (i.e., read access). If so, access is granted; if not access is denied.

4. MAC Access Isolation

An additional form of data integrity, access isolation, can be achieved by judicious use of mandatory access control levels. By setting up a label hierarchy such that user defined labels are disjoint (i.e., do not dominate) from system defined labels, the system is partitioned such that users are prohibited via MAC from reading, modifying, or executing sensitive system files, and administrators are protected from inadvertently executing untrusted code. The following picture illustrates how such a lattice may be defined:

Access Isolation Mechanism



In the lattice depicted above, the levels USER_PUBLIC and USER_LOGIN are defined for non-administrative use. The level USER_PUBLIC is defined for non-administrative user files and commands (eg., *emacs*, databases, etc) . The level USER_LOGIN is defined for non-administrative system access; by default all non-administrative users access the system at this level. The levels SYS_PUBLIC, SYS_PRIVATE, and SYS_AUDIT are defined for administrative and system use. The level SYS_PUBLIC is defined for files/commands which are accessible to both administrators and users (eg., mail, mount, date). The level SYS_PRIVATE is defined for administrative system access and is not accessible by non-administrative users. The level SYS_AUDIT is reserved for storage of the system audit trail.

Considering the lattice defined above, the commands *date* and *mail* would be labeled at SYS_PUBLIC. Since both the user and system partions have read access to data labeled at SYS_PUBLIC, both administrators and users have execute permission for these commands. Since the user does not have write permission at the SYS_PUBLIC level (MAC restricts write access), a user cannot plant a trojan horse at this level. Note that since the level SYS_PRIVATE dominates SYS_PUBLIC, the administrator does not require either mandatory or discretionary override privilege to access these files. Thus the administrator executing these commands does not have mandatory access control override permissions and therefore may only execute commands and read files at levels which are dominated by SYS_PRIVATE. Since the administrator at SYS_PRIVATE does not dominate either USER_PUBLIC or USER_LOGIN and does not acquire the privilege required to circumvent mandatory access control, the administrator is protected from invoking trojan horse programs planted at this level by users.

4.1 Discretionary Access Control

SVR4.1ES provides two complimentary DAC mechanisms: UNIX file permission modes and TRUSIX conformant access control lists (ACLs). The UNIX file permission modes are retained from previous releases of UNIX System V for compatibility. Users already familiar with UNIX file permissions will find that this mechanism still works as expected.

The SVR4.1ES ACLs are designed to satisfy the B3 level Orange Book requirements while still retaining compatibility with the UNIX file mode scheme. The SVR4.1ES ACL mechanism allows for finer control than existing file permission bits by providing the ability for the owner of an object to grant or deny access by other users to the granularity of a single user.

For convenience, SVR4.1ES ACLs also allow specification of access rights to members of groups as defined to the system in the administrative file **/etc/group**. ACLs can also be arbitrarily large; that is, the number of ACL entries is not limited by the system. The system administrator can set the maximum number of entries per ACL by setting a tunable parameter. (Naturally, as ACLs get larger, processing gets slower, which induces a practical limit on the number of ACL entries.)

In SVR4.1ES, an ACL is associated with every file system object and IPC object. ACLs for file system objects are stored in the associated inode, the first 7 entries are stored in the inode, additional entries are stored in indirectly referenced disk blocks. ACLs for IPC objects are stored in an internal structure associated with the instantiation of the IPC object.

An ACL contains all the DAC access information for the object with which it is associated. For the sake of compatibility, file permissions are displayed as usual in the expected situations, and operations on files behave as they would be expected to on any UNIX System V-based operating system. However, in SVR4.1ES, file permission bits are actually translated into and stored as ACL entries. The ACL entries which are derived from the file owner, file owner group and other permission bits are called **base entries.** Permission can be granted or denied beyond the **base** entries by inclusion of **additional** ACL entries. A simple SVR4.1ES ACL would appear as follows (note the numbers in parenthesis are used to indicate the association between the permission bits, owner and group and the ACL. They do not appear in SVR4.1ES ACLs):

```
 (4)(5)(6)          (2)   (3)                        (1)
rwxr-xr-x+   1   fred  demo   73  Jan 6 20:27  run.sh


 #file: run.sh      (1)
 #owner: fred       (2)
 #group: demo        (3)
. user::rwx .(4) . . .
: user:larry:--x    :
: group::r-x  (5)   :     or'ing these entries provides class entry
: group:sys:---. . .:
 class:r-x
 other:r-x     (6)
```

Notes:

+ sign indicates file has an associated ACL
the class entry is always equal to the group permission bits. Thus stat'ing the file provides the maximum permission granted by the ACL

An ACL consists of the following types of entries, which must be in the following order:

- **user** entry - This entry is derived from the file owner permission bits; it contains a user ID and the permissions associated with it. There is always one entry of this type, which represents the object owner and is denoted by a null (unspecified) user ID. There may be additional unique **user** entries.
- **group** entry - This entry is derived from the file group permission bits; it contains a group ID and the permissions associated with it. There is always one entry of this type, which represents the object owning group and is denoted by a null (unspecified) group ID. There may be additional unique **group** entries.
- **other** entry - This type of entry contains the permissions granted to a subject if none of the above entries have been matched. There is exactly one of these entries in an ACL.
- **class** entry - This type of entry contains the maximum permissions granted to the file group class. There is exactly one of these entries in the ACL. The **class** entry indicates the **maximum** permission allowed by the ACL. Additionally, this entry acts as a **mask** and provides compatibility for existing applications which obtain file access permission via *stat* and attempt to change file status via *chmod*, for example:

Modification of mode bits & ACL using chmod

```
Before chmod 000        After chmod 000        After chmod 755 (re-set mode bits)

rwxr-xr-x-              ---------              rwxr-xr-x-

#file: run.sh           #file: run.sh          #file: run.sh
#owner: fred            #owner: fred           #owner: fred
#group: demo            #group: demo           #group: demo
user::rwx               user::---              user::rwx
user:larry:--x          user:larry:--x         user:larry:--x
group::r-x              group::r-x             group::r-x
group:sys:---           group:sys:---          group:sys:---
class:r-x               class:---              class:r-x
other:r-x               other:---              other:r-x
```

Refering to the example above; notice that the ACL entries for file owner, other and file group class are changed to reflect the intended setting of the permission bits (via *chmod*()). No additional ACL entries are modified. The intended effect of the *chmod* 000 is accomplished by using the file group class entry as a mask. Note that the file owner group entry was not modified by the *chmod*. This is due to the fact that the SVR4.1ES implementation treats the file owner group as an additional ACL entry.

- **default** entry - This type of entry may only exist on a directory. These entries are similar to the entries described above, except that they are never used in an access check, but are used to indicate the **user**, **group**, and **other** ACL entries that should be added to a file created within the directory.

4.2 Trusted Path

The SVR4.1ES trusted path feature is a **streams** module which ensures that the user's password is being requested by login and not by a malicious program that masquerades as a system program to gain sensitive information. The SVR4.1ES trusted path mechanism is only invoked at ***login*** time and is not directly invokable by the user.

The user invokes the trusted path and subsequently gains access to the system via a terminal using the Secure Attention Key (SAK). By default the SAK is a line drop though it can be configured by the administrator to be a character or asynchronous line condition, such as a break.

The SVR4.1ES trusted path feature works as follows:

1. A user requesting access to the system enters the SAK.
2. The system identifies the SAK before any line discipline is applied.
3. On detecting the SAK, the TCB terminates any current login session, permanently puts open connections in a state such that they can no longer be used for terminal I/O, and eventually reinitiates the login sequence.
4. If login is not completed within the login timeout period, the login program will enter a mode where login interaction cannot proceed until the SAK is entered again.

4.3 Audit

Hand in hand with the ability to penetrate system security is the ability to do so without detection. On most UNIX systems the only record of process execution is the information saved by the UNIX systems process accounting facility. While this data provides some insight as to what may have occurred on the system, it can be spoofed and does not provide sufficient granularity of data to fully determine the actions of an intruder. Additionally, existing UNIX process accounting provides no granularity, it is an all-or-nothing feature; either accounting is enabled for all (known) events, for all users or it is completely disabled. Since the recording of accounting data is done on an all-event, all-user basis, a good deal of system resources are expended; for this reasons, it is frequently not used. These shortcomings have been corrected in SVR4.1ES with the addition of system auditing. Like accounting, auditing records events which occur on the system. However, in addition to simply recording the occurrence of events, auditing also records the parameters associated with the event and the outcome of the event. Granularity is provided at both the event and user level, that is, the administrator can select specific events which will be audited and can specify the users for whom those events are audited. Since the system's audit daemon runs with a mandatory access control level which is disjoint from all defined user levels, the presence of the audit daemon (i.e., the ability to detect auditing) is undetectable by unprivileged users. SVR4.1ES provides an audit mechanism capable of recording and reporting on all security-related events that occur on the system.

All security-related events that occur on the system can be audited, including those events identified as being associated with covert channels. SVR4.1ES associates most audit events with a system call. For example the **mk_dir** and **rm_dir** events map the ***mkdir*** and ***rmdir*** system calls. Since system administrators tend to think in terms of system events, SVR4.1ES provides the concept of an event **class**. The class mechanism allows for a logical grouping of event **types**. For example, the **mk_dir** and **rm_dir** events fall into the **dir_make** class. Since auditing tends to generate large amounts of data and since an administrator may wish to select most but not all of the event types within a class, SVR4.1ES permits selection by both event **type** and **class**. Additionally the selections can be intermixed (i.e., a **class** may be selected and one or more **types** within the class may be turned off).

Since a certain sub-set of applications may wish to add records to the audit trail, the SVR4.1ES audit feature provides the ability for applications to add their own free-format records to the audit trail. Multiple site or application records may be defined. These added records can be selected and later reported using the standard SVR4.1ES selection and reporting tools.

Events which are deemed critical to the integrity of the system (i.e., events critical to the integrity of the audit trail) are always audited whenever auditing is enabled regardless of the system wide and per-user event masks. These events are called *fixed* events. Other events are auditable at the discretion of the system administrator; these are called *selectable* events.

As stated above, events may be set on either a system wide or per-user basis. System wide events are selected by the administrator with the **auditset** command. **auditset** may also be used after auditing is enabled to specify additional events to be audited or to de-select events that no longer require auditing.

Per-user audit masks may be designated for each user by using the **useradd** command. These masks are permanent - whenever auditing is enabled and the user is logged on, events specified in these masks will be audited. The set of *fixed* events along with the system wide and per-user audit masks are or'ed together to form the user's process audit mask.

Each auditable event, when audited, generates an associated audit record; collected for each event audited are a time stamp, the user identity, object name, level of the process (subject) causing the event, privileges used, an identification of the type of event, and an indication of the success or failure of the event. Other information specific to the event type is also collected. The **auditrpt** command is used to select, format and print data from the log file.

5. Summary

This paper has described several security features that provide a high degree of protection against unauthorized access, viruses, and trojan horses. In most cases, system security is compromised by exploitation of an administrative oversight such as incorrect setting of file mode bits. Meticulous use of the security features already present in the UNIX system can eliminate or greatly reduce most breaches of system security. However, since most, if not all, of the current UNIX system security features rely solely on administrator discretion, no matter how carefully a system is administered, mistakes can and do occur. When mistakes do occur, the system is left vulnerable in some area. System enforced features such as mandatory access control and least privilege eliminate or greatly reduce the amount of compromise that can occur if an administrative flaw is detected and exploited. Thus the burden of system protection is no longer solely dependent on the administrator.

6. REFERENCES

[1] Department of Defense. *Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December, 1985.

[2] Bell, D. E. and LaPadula, L. J. *Secure Computer System: Unified Exposition and Multics Interpretation*, MITRE Corporation, MTR-2997, March 1976.

KNOWLEDGE-BASED COMPUTER SECURITY ADVISOR*

W. J. Hunteman and M. B. Squire
Safeguards Systems Group, MS-E541
Los Alamos National Laboratory
P.O. Box 1663
Los Alamos, NM 87545

ABSTRACT

The rapid expansion of computer security information and technology has included little support to help the security officer identify the safeguards needed to comply with a policy and to secure a computing system. Los Alamos is developing a knowledge-based computer security system to provide expert knowledge to the security officer. This system includes a model for expressing the complex requirements in computer security policy statements. The model is part of an expert system that allows a security officer to describe a computer system and then determine compliance with the policy. The model contains a generic representation that contains network relationships among the policy concepts to support inferencing based on information represented in the generic policy description.

I. INTRODUCTION

The field of computer security is continuing to expand the information security technology available to address security concerns in computing systems. The advances are often directed towards technological solutions of a multidimensional problem, but the nontechnical areas have received little, if any, serious effort towards improving the entire security environment surrounding a computing system. The use of trusted computing systems alleviates the problem somewhat by implementing the nondisclosure policy in a standard manner [1]. However, this approach does not address other equally important security issues such as other policy components (e.g., personnel security and physical security) or the interaction between the Trusted Computing Base (TCB) and the security features in the local environment (e.g., administrative procedures).

This paper describes an effort initiated at Los Alamos to create a knowledge-based system to act as an "expert" Advisor to a security officer. The Advisor will consider the total environment, including policy requirements, when identifying the security needs for a computing system. The Advisor provides an automated capability to support the system certification process. System certification, as described in References 2 and 3, requires an analysis of the system security features, threats against the system, and the system operating environment according to an information security policy. The Advisor system is designed to be used during the development of a secure system and when reviewing or certifying the security of an existing system for compliance with a policy.

Most policy statements are complex and difficult to interpret for a local computing system environment. This difficulty generally arises from the desire for the policy to allow the maximum flexibility for changes in the hardware or software configurations of a computing system. Experts from the policy-making organizations will also sometimes give conflicting advice regarding policy implementation for a particular system. The lack of clear guidance on applying the policy and the absence of a consistent approach to implementation suggest that a uniform methodology is needed to aid the security officer in interpreting and applying security policies.

*Work supported by the US Department of Energy, Office of Safeguards and Security.

The methodology being developed as part of the Advisor provides a consistent decomposition and interpretation of policy statements into a knowledge base that can be used to guide the selection of safeguards for a specific computing system. The Advisor architecture is designed to

- support the semantic or conceptual representation of a complex system;
- if appropriate, collect and organize information about local or site-specific policies;
- support automated reasoning about the represented system;
- manage the use of uncertain or incomplete information in the knowledge networks;
- support "what-if" experimentation to adjust the local environment implementation description; and
- provide, on request, justification or explanation of each decision throughout the process.

The architecture supports multiple representations of policies, regulations, local or site-specific implementations of the policies, and the interdependencies between the various concepts and implementations. The architecture is designed to allow the development of user oriented interfaces that display information in a manner consistent with the user's vocabulary and operating environment. The Computer Security version of the Advisor will implement the Department of Energy (DOE) Classified Computer Security Program defined in DOE Order 5637.1 [3].

II. POLICY REPRESENTATION ISSUES

A policy statement is intended to guide personnel in constructing a local environment that has some general property, such as a safe or secure environment. Policy statements are usually written by, or with the help of, experts in the field. Policy implementors, however, often lack the complete understanding to interpret the exact meaning of the policy. Some statements may be unclear, such as "Procedures for identifying and authenticating users must be addressed." This may be either an oversight by the policy writer or a deliberate ambiguity to allow flexibility of interpretation. If it is for flexibility, the implementor must decide how to interpret the intent and then implement a solution. Typically this solution must then be approved by an approval or accrediting authority who may have a different interpretation of the policy. Some organizations also allow implementors to create unique interpretations and implementations of the policy requirements, subject to approval by the accrediting authority. Regardless of the allowed flexibility, there are some characteristics that seem to be shared by all policy statements.

A. Property/Requirement Coupling

When a policy is broken down into specific requirements, the requirements can be expressed as a coupling of a specific problem and the expected solution. These requirement/solution pairs can be viewed as a list of IF/THEN statements. For example, a policy statement could be

IF a computer processes classified information
THEN it must have identification and authentication procedures.

We call the IF clause a property, and the THEN clause a requirement. A property is the activity or condition that must be present or practiced to meet the requirement. We refer to this coupling of property and requirement as a property/requirement (p/r) couple. Most instances of a p/r couple can be further decomposed. The property can be expressed as a nested set of conjunctions and disjunctions of objects, relations, and attributes. Similarly, the requirement can also be expressed as a nested set of conjunctions and disjunctions.

B. Existence/Event Coupling

Policy statements also have a distinction between passive p/r couples and active p/r couples. A passive policy statement does not explicitly or implicitly require invoking a specific requirement based on some action by a subject, such as a user or process. For example, the following could be viewed as a passive p/r couple because there is no explicit requirement to invoke the requirement.

IF a computer processes classified information
THEN it must have identification and authentication procedures.

However, in most policies there is either an explicit or implied requirement to respond to action by a subject. For example, the implied active part of the policy statement in the above example, could be

IF a subject attempts to logon to a computer
THEN identification and authentication procedures must be invoked.

We refer to the passive part of this policy element as the existence and to the active part as the event. These pairs of existence and event p/r couples are referred to as existence/event (e/e) couples. It is possible that either the existence or the event could be empty. For example, there is no related event p/r couple in the following:

IF a computer processes classified information
THEN it must be in a protected area.

Property/requirement couples based on events are slightly more complicated and can be modelled as state changes in the policy knowledge network. Many policies require that certain procedures be done periodically. These can be modelled as an event, namely, the passage of time. For example, it may be required that a computer system is reviewed annually. This can be modelled as the event of a year passing or a time-related transition.

Problems based on existence will be referred to as "vulnerabilities," and solutions based on existence as "safeguards." We will refer to problems based on events as "attacks" and to solutions based on events as "responses." An interesting property of most policy statements is that whenever an existence problem occurs, then the expected solution is also based on existence. Similarly, problems based on events have solutions based on events.

C. Hierarchical Order of Policy Statements

Policy statements are often hierarchically arranged. First, the e/e couples can be arranged by some categorical hierarchy. For example, all e/e couples relating to "Personnel Security" can be grouped into one category, which in itself can be a category in "Computer System Security." Also, each property or requirement can be composed of subproperties/subrequirements. The subproperties/subrequirements can also be further refined with the subordinate items categorizing and defining their parents. Figure 1 depicts the general representation of a policy element used by the Advisor model.

D. User Defined Solutions

Some policies allow users to develop their own solutions to policy requirements. This approach effectively allows the user to modify the hierarchy under the requirement part of one or more p/r couples. Often the security officer is allowed to create a specific solution to the problem as long as it satisfies the general intent of the policy. The Advisor model allows a controlled capability for security officers to substitute approved alternative solutions.

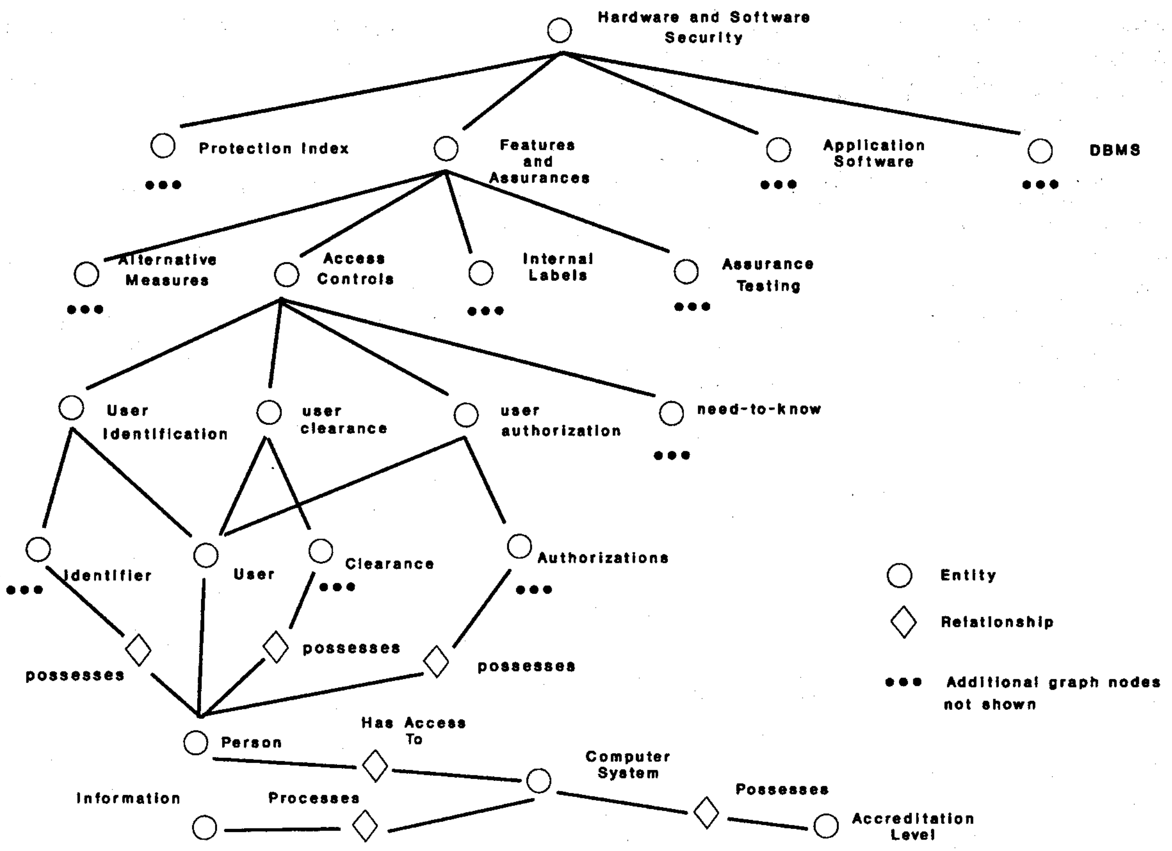


Figure 1. Conceptual graph of policy fragment.

III. POLICY REPRESENTATION

A. Policy Representation Requirements

An acceptable representation of a policy statement must be able to represent the domain addressed by the policy, differentiate between the policy concepts and instances, and support a categorical organization of the policy. First, we must be able to accurately represent the policy domain. In addition to properties and requirements, we must be able to represent relationships between properties and requirements, interactions between events and p/r couples, and time. For example, suppose we wished to represent a personnel security policy for a secure computer system. We must be able to represent such concepts as computers, classification levels, and users. We must also be able to represent relationships between these concepts, such as the relationship between a computer and its users. We also must be able to differentiate between instances and concepts. For example, if the policy states that all classified computers must be in protected areas, we want to be able to differentiate between the concept of a classified computer and a particular instance of a classified computer. The representation approach must also support a categorical hierarchy for the e/e couples. The Advisor model also allows for controlled modifications to the hierarchy when the policy supports implementor flexibility. The user modifications are restricted to properties already defined in the policy domain. For example, if the policy allows substitution of physical protection for user identification and authentication, then the user must be restricted to selection of known and approved physical protection properties when making the modification. Events must also be represented. For example, we must be able to represent the event of a user login to the computer. We must also be able to model procedures, such as the generation and distribution of authenticators.

B. Advisor Model Representation

The Advisor model uses conceptual graphs [4] to represent policy information. The policy representation conceptual graph contains three types of nodes: category, policy, and network. Category nodes are used to organize the high-level segments of the policy. Policy nodes represent e/e couples. A policy node may be connected to up to four network nodes. Network nodes represent the policy node's existence p/r couple and its event p/r couple. Network nodes are the clauses of the IF/THEN structures. The generic representation of policy nodes and network nodes is given in Figure 2.

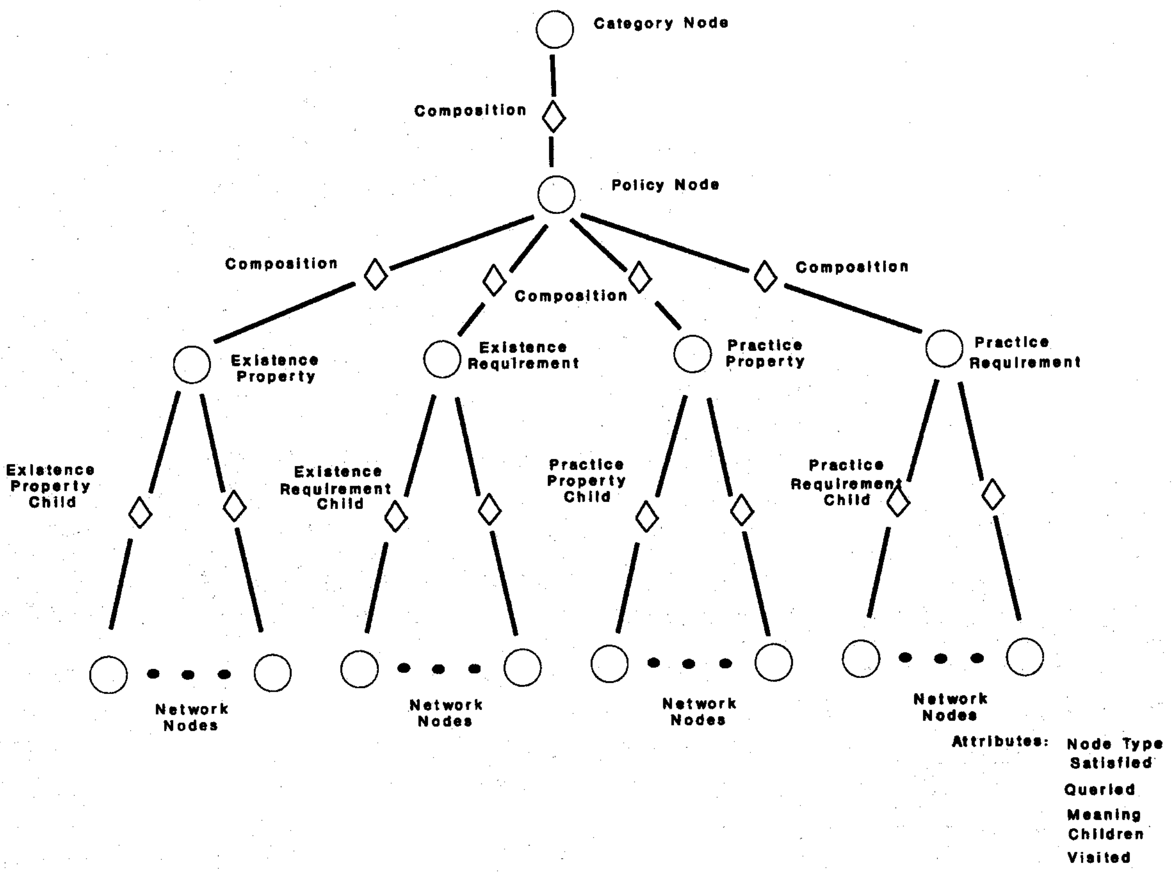


Figure 2. Generic Advisor model.

C. Advisor Architecture

The Advisor architecture, shown in Figure 3, contains two different networks [5]. The Computing Environment network is composed of network nodes that are used to guide and collect user-supplied descriptions of the local computing environment (instantiations). The Policy network contains category, policy, and network nodes that represent the policy. The Analysis component is software that evaluates the instantiations against the policy and reports the results. The Developer Interface contains facilities for creating and maintaining the Policy and Computing Environment Networks. The User Interfaces provide capabilities to allow the user to enter, view, and manipulate information in a user-friendly manner.

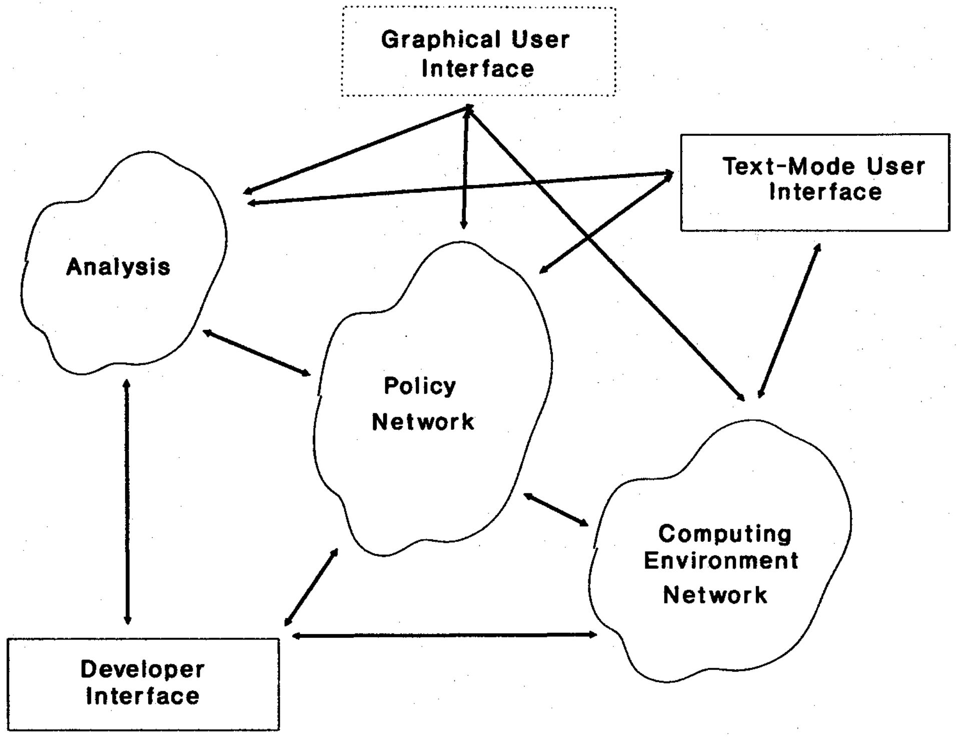


Figure. 3. Advisor architecture.

The Policy network also supports a global analysis of the policy statement by supporting the representation of multiple policies and networks of attacks/vulnerabilities and responses/safeguards that represent everything the policy must address. If a node in the Policy network cannot be associated with an attack, then either the attack/vulnerability network is incomplete and must be expanded, or that property or requirement is superfluous and should not be in the policy statement. If there is an attack/vulnerability that does not match any property or requirement, then this attack/vulnerability is not addressed by the policy, indicating an incompleteness of the policy statement. A similar analysis can be performed with responses/safeguards.

D. Advisor Knowledge Network

The interior nodes of the knowledge network may be either AND, OR, or XOR (exclusive or) nodes. Each interior node will have a node-type attribute (either AND, OR, or XOR), a satisfied attribute (YES/NO), and a meaning attribute. AND nodes require that all of their children be addressed during instantiation and analysis. OR nodes represent redundant information and allow any of their children to be addressed during instantiation and analysis. XOR nodes are used when policy elements conflict with each other and only one child will be considered during instantiation and analysis. An example of conflicting policy elements might be an audit trail that recorded every keystroke entered by a user and normal password security. The complete audit trail would contain user- and file-access passwords, while password security would not allow the passwords to be exposed in a clear form.

The satisfied attribute specifies whether or not the user has supplied the information for an instantiation of this node and whether or not the node is satisfied by the instantiation. An AND node is considered satisfied only if all of its children are satisfied. An OR node is considered satisfied if any of its children are satisfied. An XOR is considered satisfied if one of its children is satisfied and the other is not. If both children of an XOR node are satisfied a conflict is reported to the user.

The meaning attribute is used with network nodes in the Policy network to provide a linkage between the Policy and Computing Environment networks. The meaning attribute contains the name of a node in the Computing Environment network that is expected to contain a user-supplied instantiation. During the analysis phase, the Policy network is searched for the meaning attribute strings that are used to extract the instantiations for further analysis in the Policy network.

E. System Evaluation

The leaf nodes of the Computing Environment network contain the user provided instantiations and allow the Advisor to query the Policy network to determine if a p/r couple is satisfied. This information on the satisfied attribute of the child is then used by the parent concept to determine whether or not it is satisfied. A leaf node in the Computing Environment network is considered satisfied if an instantiation for the concept has been provided by the user. The information on the satisfied attribute of the leaf node is propagated to the top of the Computing Environment network where it is used to determine if the parent p/r couple is satisfied. The satisfaction of a p/r couple is then used to determine the satisfaction of individual policy couples in the Policy network.

IV. USER INTERFACE

The Computer Security Advisor implementation is designed to support the needs and activities of all of the positions identified in the Department of Energy (DOE) Classified Computer Security Program. These positions include Computer Security Program Manager (CSPM), Computer Security Operations Manager (CSOM), Computer Security Site Manager (CSSM), and Computer System Security Officer (CSSO). The CSPM is responsible for establishing the classified computer security policy for the DOE. The CSPM is also responsible for developing and maintaining a definition of the threats to DOE and contractor facilities. The CSPM may, under certain circumstances, be an accrediting authority for complex computer systems or systems that cross CSOM responsibility boundaries. The CSOM position is typically assigned to an individual in the DOE Operations Office and is responsible for oversight and guidance of the computer security program implemented by the Operations Office and any DOE contractors reporting to the Operations Office. The CSOM is responsible for review and approval of ADP Security Plans for all computer systems processing classified information in the DOE office or contractor facilities. The CSOM is typically the accrediting authority for these computer facilities. The CSSM is the individual responsible for the classified computer security program at the site or facility. The CSSM is the principal contact point and coordinator for all communications and interactions between the site and the CSOM. The CSSM is responsible for review of all ADP Security Plans and the certification of the computer systems during the accreditation process. The CSSM is also responsible for defining and implementing site-wide computer security procedures. The CSSO is the security officer responsible for defining and implementing the computer security procedures and mechanisms for a computer system that processes classified information. The CSSO is also responsible for generating and maintaining the ADP Security Plan and the ADP Security Test Plan.

The user interface of the Computer Security Advisor is based on the windowing system supported by Sun Microsystem's Open Look software. The user is presented with a series of successively detailed windows that are oriented to the particular function requested by the user.

The initial window, Figure 4, allows the user to select the desired interaction level (security officer, reviewer, or developer).

The security officer window, Figure 5, allows the user to select operations to load or save the Policy and Computing Environment networks (FILE button), exit the Advisor (QUIT button), edit or display the Policy and Computing Environment networks (EDIT button), describe a computer system (CREATE SYSTEM button), or evaluate the described system against the policy requirements (ANALYSIS button). The ANALYSIS and CREATE

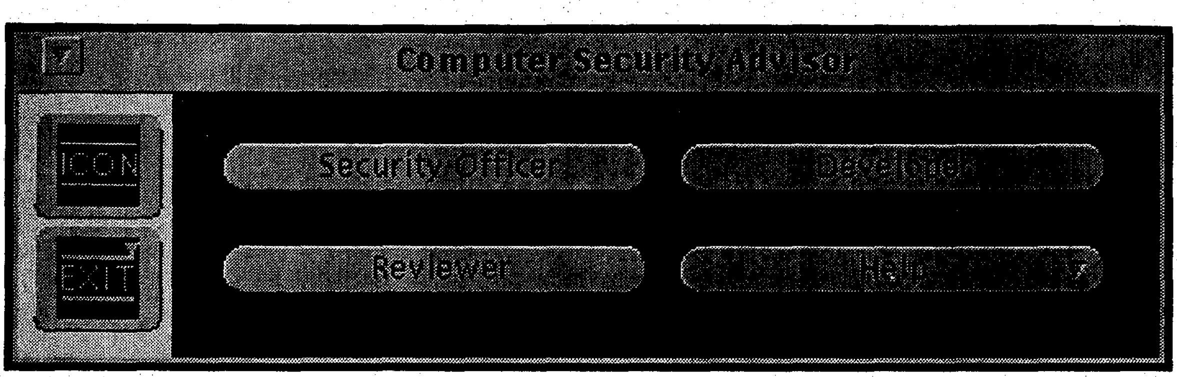


Figure 4. Initial Advisor screen.

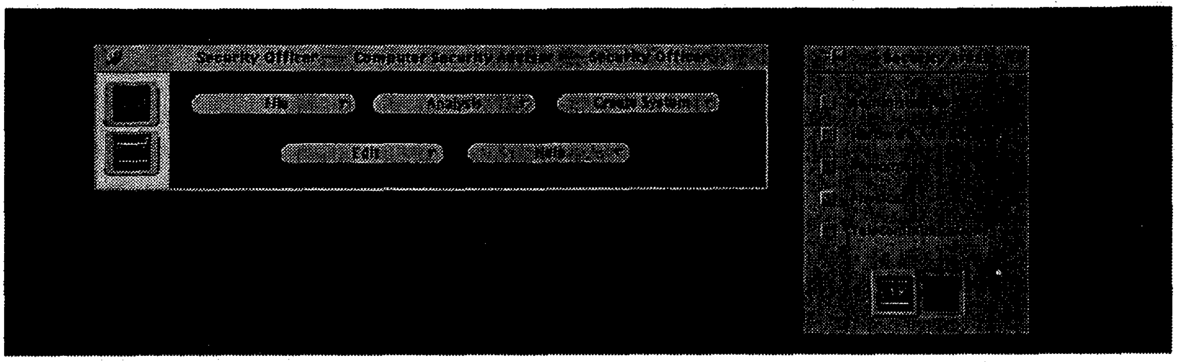
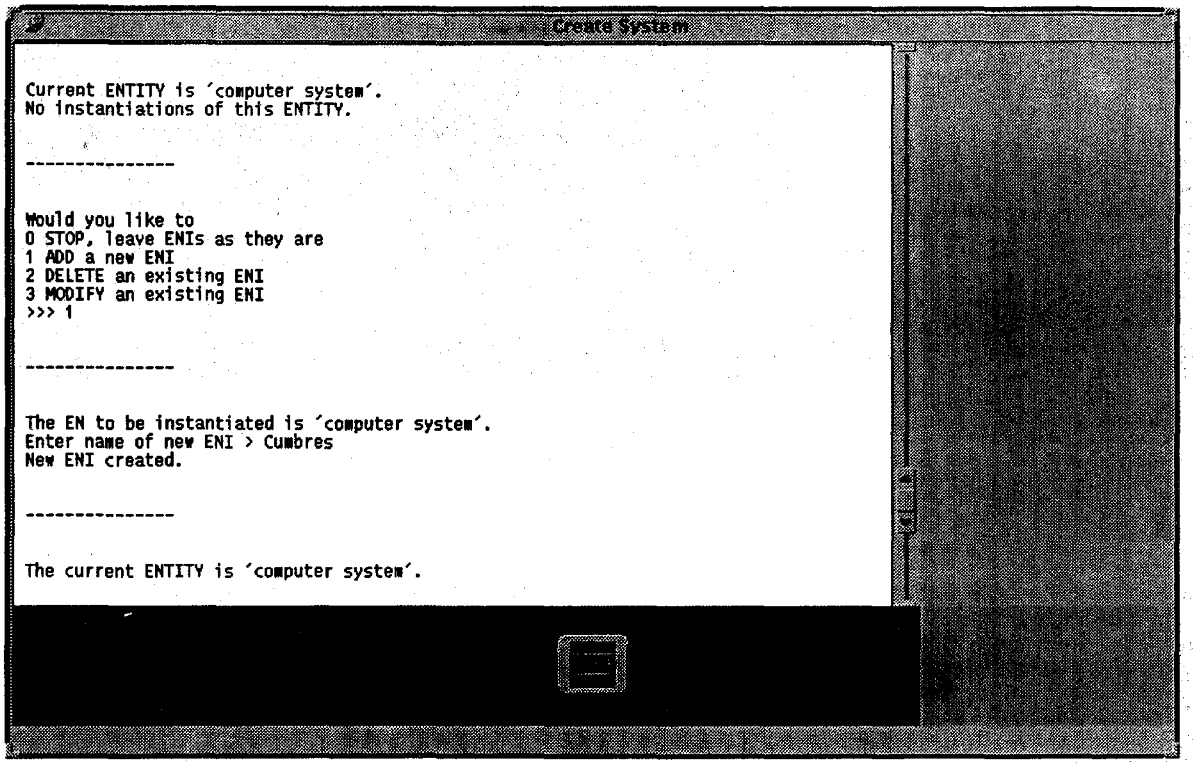


Figure 5. Initial security officer screen.

SYSTEM functions allow the user to analyze or describe the entire system environment or select a specific subset of the environment defined by the policy network. The DOE policy is divided into personnel security, physical security, telecommunications security, administrative security, and hardware/software security sections.

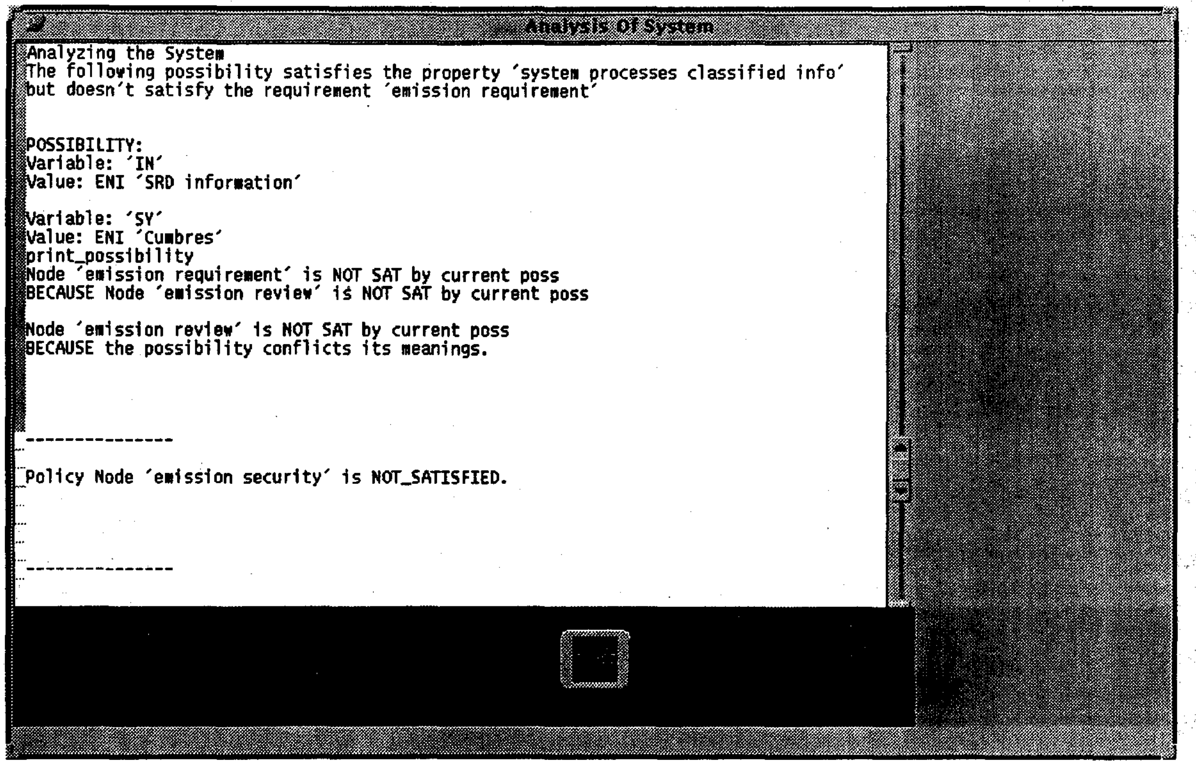
The CREATE SYSTEM functions guide the user through the process of specifying the instantiations of the computer system. The Advisor searches the Computing Environment network for concepts that must be instantiated to satisfy the policy. When a required concept is found, the user is asked to respond if the concept is present or practiced in the local environment. If appropriate, the user is also asked to identify the instance (e.g., name or procedure title). Figure 6 contains an example of the instantiation activity.

The ANALYSIS functions initiate the evaluation of the computer system against the policy requirements. After the evaluation is completed, the results are displayed for the user. Figure 7 contains a sample display showing the results of an analysis. If all p/r couples in the Policy network are satisfied, then only a single line is displayed stating that the top level policy network node was satisfied. If one or more p/r couples are not satisfied, then the unsatisfied p/r couple(s) are displayed with all subordinate p/r couples that contributed to the failure of the top level p/r couple.



```
Current ENTITY is 'computer system'.
No instantiations of this ENTITY.

---------------

Would you like to
0 STOP, leave ENIs as they are
1 ADD a new ENI
2 DELETE an existing ENI
3 MODIFY an existing ENI
>>> 1

---------------

The EN to be instantiated is 'computer system'.
Enter name of new ENI > Cumbres
New ENI created.

---------------

The current ENTITY is 'computer system'.
```

Figure 6. Instantiation window.



```
Analyzing the System
The following possibility satisfies the property 'system processes classified info'
but doesn't satisfy the requirement 'emission requirement'

POSSIBILITY:
Variable: 'IN'
Value: ENI 'SRD information'

Variable: 'SY'
Value: ENI 'Cumbres'
print_possibility
Node 'emission requirement' is NOT SAT by current poss
BECAUSE Node 'emission review' is NOT SAT by current poss

Node 'emission review' is NOT SAT by current poss
BECAUSE the possibility conflicts its meanings.

---------------

Policy Node 'emission security' is NOT_SATISFIED.

---------------
```

Figure 7. Analysis window.

V. IMPLEMENTATION

The Computer Security Advisor prototype is implemented on a Sun Microsystems workstation in C. The Advisor uses the KNET library from KONEXSYS Corporation to manage the Policy and Computing Environment network space. The user and developer interfaces are implemented in the Open Look windowing environment provided by Sun Microsystems.

VI. SUMMARY

A knowledge-based system has been developed to collect and organize knowledge from computer security experts for use by a security officer. The Advisor includes a model that incorporates all aspects of a policy statement. The Computer Security Advisor contains a generic description of the desired policy and the user interface to support a security officer description of the local system and analysis of policy compliance. The system is policy-based and contains the flexibility needed to support changes in policies and hardware and software technology.

REFERENCES

1. "Department of Defense Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, National Computer Security Center (1985).

2. National Research Council, *Computers At Risk: Safe Computing in the Information Age* (National Academy Press, Washington D.C., 1990).

3. "Classified Computer Security Program," Department of Energy Order 5637.1, Department of Energy (January 1988).

4. J. F. Sowa, *Conceptual Structures–Information Processing in Mind and Machines* (Addison-Wesley, Massachusetts, 1984).

5. N. V. Findler, Ed., *Associative Networks* (Academic Press, New York, 1979).

THE LOGISTICS OF DISTRIBUTING A SMART TOKEN

Dawn A. Brown
Department of Defense
Ft. George G. Meade, MD 20755-6000
(301) 859-4360

ABSTRACT

This paper will address the logistics of distributing a smart token on a computer system. A smart token is an identification and authentication device for a host computer system. This paper will address the logistics from four perspectives. The first perspective will discuss why the smart token, WATCHWORD Generator was implemented on DOCKMASTER. A cost analysis, including procurement of the smart token, batteries, man hours, and maintenance is the second perspective. The third perspective discusses how the smart token will enhance the security of the host computer system. How DOCKMASTER will respond when a user is trying to access the system with the WATCHWORD Generator implemented is the fourth perspective. With a successful method of identifying and authenticating users of the computer system, the system is less susceptible to penetration.

INTRODUCTION

DOCKMASTER is the National Security Agency's (NSA) unclassified computer system that directly supports the missions and functions of the National Computer Security Center (NCSC). DOCKMASTER was established as the Information Security Showplace for dissemination and exchange of Information Security data. DOCKMASTER executes the Multics Operating System which was granted a B2 security rating based on the guidelines defined in the Department of Defense Trusted Computer System Evaluation Criteria, also known as the "Orange Book".

With the increasing number of computer penetrations, it is vital that each computer user is correctly identified when accessing a computer system. The process of correctly identifying each computer user is called authentication. The primary authentication device on DOCKMASTER is the WATCHWORD Generator.

The WATCHWORD Generator is a portable, hand held authentication device that is used in conjunction with the user's password during the login process. Each WATCHWORD Generator is assigned a unique Personal Identification Number (PIN) and Secret Key. During the login process, the user must correctly authenticate his/her login process by using the WATCHWORD Generator. The WATCHWORD Generator will generate a different response during each login process based on the "Challenge" generated from DOCKMASTER. If the correct response to the "Challenge" is not entered the user will be denied access to DOCKMASTER.

WATCHWORD GENERATOR IMPLEMENTATION

With every computer system there should be a means of authenticating who is trying to access the system. As with most computer systems, DOCKMASTER uses the userid and password option as a means of authenticating each user. However, should this option be the only means of authenticating users? The answer depends on several questions. For example, what type of data (unclassified, classified, proprietary) is the user trying to access, should the user have access to this data, and is the data restricted to specific users. If the answer to any of these questions is yes, then the userid and password option should not be the only means of authenticating users.

In 1987, DOCKMASTER Management was faced with the question, how can we enhance the protection of restricted data while also authenticating each user. A decision was made to add an additional layer of security to the login sequence that would identify and authenticate each user requesting access to restricted data. A month long operational test consisting of twenty-one users accessing DOCKMASTER through various methods (Direct Dial, Tymnet, Telnet, etc.) was conducted. Based on the conclusions of the test, the WATCHWORD Generator was chosen as the most effective way to add the additional layer of security to DOCKMASTER.

COST ANALYSIS

There are overhead costs involved in the implementation and use of the WATCHWORD Generator. Some of the overhead costs include:

a. The WATCHWORD Generator software.
b. The WATCHWORD Generator devices.
c. The WATCHWORD Generator batteries.
d. The WATCHWORD Generator Administrator duties.
e. Maintenance and recovery of the WATCHWORD Generators.
f. Replacement WATCHWORD Generators and batteries.

The initial overhead cost of the WATCHWORD Generator includes procuring the software for the WATCHWORD Generators. This software is necessary to communicate with the host computer. Additionally the cost of one device for each user that requires authentication by the system must be incurred. The WATCHWORD Generator costs approximately ninety dollars each. Given a user population of five hundred, the total cost to procure the WATCHWORD Generator is approximately forty-five thousand dollars. This figure may appear to be substantial at the outset, but consideration should be given to the thousands of dollars that will be saved when the WATCHWORD Generator is implemented.

When a computer system is compromised, time and money must be spent on tracing the path of the computer hacker, notifying users of the penetration so that they can change their passwords and ensure that their data was not compromised, and investigating why the penetration occurred. The cost involved in this whole process can be substantial. The time and money that must be invested if the computer system is compromised will not have to be incurred if the WATCHWORD Generator is implemented. The chances of a computer system being compromised with the WATCHWORD Generator implemented is virtually zero. The advantages of implementing the WATCHWORD Generator out way the disadvantages considerably.

The WATCHWORD Generator is battery operated, thus the cost of the batteries is a second overhead cost. Each WATCHWORD Generator requires two calculator or equivalent batteries. The cost per set of batteries for the WATCHWORD Generators is less than one dollar. As with the cost of the WATCHWORD Generator device, the cost of the batteries is minute compared to the advantages and additional security that the WATCHWORD Generator will bring to the computer system.

The third overhead cost includes the actual man hours involved in implementing the WATCHWORD Generator. Every computer system should have one or more individuals that concentrates on the security of the system. This person is usually called the Computer Security Officer (CSO). The CSO may be a prime candidate to implement the WATCHWORD Generator since the WATCHWORD Generator does add an additional layer of security to the computer system. However, the CSO does not have to implement the WATCHWORD Generators,. A WATCHWORD Generator Administrator (WGA) should be appointed.

The WGA responsibilities should include, but are not limited to, installing batteries into the WATCHWORD Generator device, assigning a unique PIN to each device, keying each device with a unique secret key, recording each device in the controllers and database, maintaining an accurate inventory of WATCHWORD Generators and batteries, and ensuring the return of unused WATCHWORD Generators for reissuance.

Each device requires approximately fifteen minutes to implement on the computer system. Based on the number of devices that will be implemented at one time, the number of man hours invested is also minimal. The relatively small number of man hours invested is small price to pay for the numerous advantages that implementing the WATCHWORD Generator will provide.

Ensuring the return of unused WATCHWORD Generators may require the most man hours. For example, if a user changes job positions, relocates, is fired, or if the company moves, it is the responsibility of the WGA to locate the user and ensure the return of the WATCHWORD Generator. A Standard Operating Procedure (SOP) should be established to deal with problems such as the ones listed above. With a well defined SOP the WGA should not have any problems in deciding what the next step should be in ensuring the return of the WATCHWORD Generators.

The life span of the batteries for the WATCHWORD Generators is approximately two years. Therefore, to minimize user inconvenience, a system of exchanging WATCHWORD Generators must be implemented. The WGA must issue each user a new WATCHWORD Generator. Each WATCHWORD Generator must have a new PIN as well as a new secret key. The purpose of issuing a new PIN and secret key is to enhance key management and security of the computer system.

During the exchange phase of the WATCHWORD Generators, each user will have two WATCHWORD Generators for a short period of time, but only one WATCHWORD Generator will be used to authenticate the user. The WGA must explain to the user population the procedures of why, when, and how the replacement WATCHWORD Generator will be used. This process can become extremely confusing if a detailed plan is not implemented. The exchanging of WATCHWORD Generators will enhance the security of the computer system by reducing the chances of a users PIN and or secret key being compromised. The longer a user utilizes the same PIN the greater the possibility that their PIN will be compromised.

Some may argue that it would be easier and less time consuming to issue new batteries to each user. This would not be a feasible method because once the batteries are removed the memory is automatically erased. Once the PIN and secret key is erased, the device will no longer be able to function as a smart token.

The cost involved in the exchange process is also minimal. If an adequate number of WATCHWORD Generators and batteries are procured during the initial phase, the only cost that should occur is the cost of mailing the replacement WATCHWORD Generators and the man hours to implement the exchange process.

WATCHWORD GENERATOR AND SECURITY

To reiterate, the implementation of the WATCHWORD Generator can only enhance the security of the computer system. Some of the enhancements include, as a minimum:

a. Providing the user community with a secure processing environment.
b. Identifying and authenticating each user to ensure that they have access to information they need.
c. Restricting sensitive data to only specific users who have access to review such data.
d. Providing an extra layer of security for the user and the computer system in the event that the password is compromised.
e Reducing the probability that the computer system will be compromised.

Each user is assigned a unique PIN and secret key, however, the secret key is not known to the user. The secret key is entered into the WATCHWORD Generator by the WGA before it is issued to the user and is not accessible by the WATCHWORD Generator. Because each PIN and secret key is unique for each WATCHWORD Generator, a computer hacker would have to physically have the WATCHWORD Generator, userid, password, and PIN of the user whom account he/she is trying to compromise.

DOCKMASTER LOGIN WITH THE WATCHWORD GENERATOR

When a DOCKMASTER user logs in with the WATCHWORD Generator the sequence of identification and authentication begins. After the user enters his/her userid and password, DOCKMASTER will "Challenge" the user for a response. At this point the user must enter his/her PIN into the WATCHWORD Generator followed by the seven-digit system "Challenge". The WATCHWORD Generator will generate a seven-digit "Response" that the user will enter into DOCKMASTER. If the user has correctly entered in his/her userid, password, PIN, Challenge, and Response, DOCKMASTER will allow the user access to the system. If any of the above elements were entered incorrectly, DOCKMASTER will not grant access to the system.

If the PIN is entered incorrectly, the secret key will be unable to generate a correct response to the "Challenge". Although a "Response" will be generated, it will not be correct, therefore the user will not gain access to DOCKMASTER. Also if the "Challenge" is entered incorrectly into the WATCHWORD Generator, a "Response" will be generated for that "Challenge" not the system generated "Challenge". Since the wrong "Challenge" was entered, thus generating an incorrect "Response", DOCKMASTER would deny the user access to the system.

FUTURE OF THE WATCHWORD GENERATOR ON DOCKMASTER

The WATCHWORD Generator has been an overwhelming success on DOCKMASTER. Although the implementation of the WATCHWORD Generator on DOCKMASTER caused minimal user frustration, the majority of the DOCKMASTER user population view the implementation as a positive step toward better computer security.

Where do we go from here? There are two options that the WATCHWORD Generator offer that can be utilized by the DOCKMASTER user community. The first option includes user authenticating login to DOCKMASTER. The user can send a "Challenge" to the host computer, DOCKMASTER, and the host computer will generate a "Response". If the correct "Response" is given, the user will know that he/she is logging into the correct computer system.

The second option includes issuing the user two PINs and secret keys. The WATCHWORD Generator has the capability of storing two PINs and secret keys for user identification and authentication. This option will add another step to the identification and authentication sequence as well as enhance security. This option would be excellent for System Administrators. Because of the privileges that System Administrators have, this option would greatly decrease the chances of a computer hacker compromising a System Administrator's account.

Although neither of the options are being implemented on DOCKMASTER in the near future, the options still remain open. Before either option is implemented, a need assessment will be thoroughly conducted and based on the conclusions the options may or may not be implemented.

CONCLUSION

With the growing concern for computer security, the implementation of the WATCHWORD Generator on DOCKMASTER has greatly reduced the chances of the system being compromised. Although no system is one hundred percent capable of preventing a successful penetration, the WATCHWORD Generator does provide that extra layer of security.

The advantages of implementing a smart token on a computer system outweighs the disadvantages considerably. Providing a secure processing environment for computer users is one of the the main concerns of computer security and the implementation of a smart token would be a step in the right direction for ensuring computer security.

✰ U.S.GOVERNMENT PRINTING OFFICE:1991-525-425/40669