

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY/

NATIONAL COMPUTER SECURITY CENTER

18th NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE

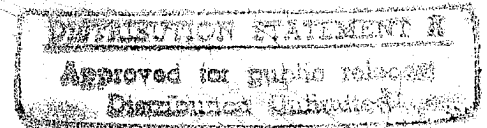
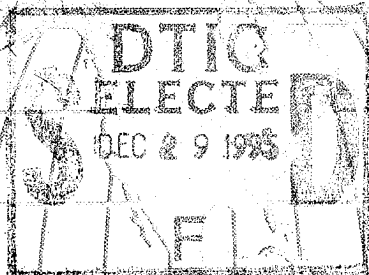
(formerly the National Computer Security Conference)

19951228 004

October 10-13, 1995

Baltimore Convention Center

Baltimore, Maryland



PROCEEDINGS VOLUME I

Making Security Real

Welcome

The National Computer Security Center (NCSC) and the Computer Systems Laboratory (CSL) are pleased to welcome you to the Eighteenth National Information Systems Security Conference. The new conference name reminds us that information systems, not just computers, must be secure. This year's program, with its theme "Making Security Real," is designed to help you plan for effective use of information security technology and to create security solutions. We believe the conference will stimulate a copious information exchange and promote a solid understanding of today's information security issues and protection strategies.

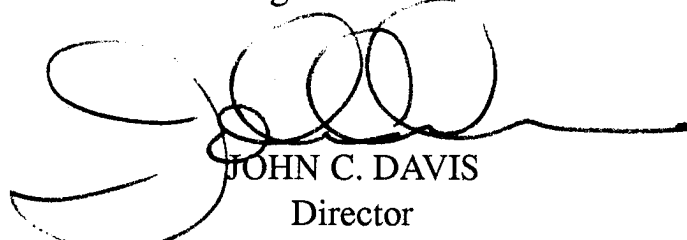
The conference program addresses a wide range of interests from technical research and development projects to user oriented management and administration topics. This year, the program focuses on developing and implementing secure networks, technologies, applications, and policies. Papers and panel sessions address a broad spectrum of network security subjects including: security architecture, internet security, firewalls, multilevel security (MLS) products, and security management. Because the National Information Infrastructure (NII), and its present backbone--the Internet--are topics of increasing interest, the challenges they present are the subject of many presentations. As in the past, a number of tutorials introduce attendees to a variety of information security topics and project areas. As a new feature this year, we have invited the vendor award recipients to provide product information displays as part of the award ceremony.

We feel assured that the professional contacts that you make at this conference, the presentations, and these Proceedings will offer you insights and ideas you can apply to your own security planning efforts. We encourage you to share the ideas and information you acquire this week with your peers, your management, and your customers. We also encourage you to share with us your success-based security techniques. It is through sharing that we will continue to enhance the security of our information systems and networks and build a strong foundation to make security real.



SHUKRI A. WAKID

Acting Director
Computer Systems Laboratory



JOHN C. DAVIS

Director
National Computer Security Center

DTIC QUALITY INSPECTED 3

18th National Information Systems Security Conference

Co-Chairs

Stephen F. Barnett, *National Computer Security Center*
Irene Gilbert Perry, *National Institute of Standards and Technology*

Program Directors

Jack Holleran, *National Computer Security Center*
Dennis Gilbert, *National Institute of Standards and Technology*

Program Council

Edward Borodkin, *National Computer Security Center*
Christopher Bythewood, *National Computer Security Center*
Dr. Gary Smith, *Arca Systems, Inc.*

Administration

Stacey Duany, *National Computer Security Center*
Tammie Grice, *National Institute of Standards and Technology*
Mary Groh, *National Computer Security Center*
Barbara Keller, *National Security Agency*
Kathy Kilmer, *National Institute of Standards and Technology*
Carol O'Brien, *National Computer Security Center*
Phyllis Pierce, *National Computer Security Center*
Pat Purkey, *National Security Agency*
Sara Torrence, *National Institute of Standards and Technology*

Conference Referees

Dr. Marshall D. Abrams
Rowland Albert
James P. Anderson
Devolyn Arnold
Al Arsenault
Steve Barnett
Dr. Matt Bishop
Earl Boebert
Edward Borodkin
Dr. Dennis Branstad
Dr. Martha Branstad

The MITRE Corporation
National Security Agency
J. P. Anderson Company
National Security Agency
National Security Agency
National Computer Security Center
University of California, Davis
Sandia National Laboratory
National Computer Security Center
Trusted Information Systems, Inc.
Trusted Information Systems, Inc.

Conference Referees *(continued)*

Dr. Blaine Burnham Christopher Bythewood Dr. William Caelli Dr. John Campbell Dr. Jon David Dr. Dorothy E. Denning Donna Dodson Ellen Flahavin Daniel Gambel Virgil Gibson Dennis Gilbert Dr. Grace Hammonds Ronda R. Henning Dr. Harold Highland, F.I.C.S., F.A.C.M. Jack Holleran Hillary H. Hosmer Dennis Huamán Dr. Sushil Jajodia Carl Landwehr Robert Lau Dr. T. M. P. Lee Special Agent John F. Lewis Steve Lipner Teresa Lunt Dr. John McLean Sally Meglathery Rebecca Mercuri William H. Murray Dr. Peter Neumann Donn B. Parker Dr. Charles Pflieger Loreto Remorca Dr. Ravi Sandhu Marvin Schaefer Dr. E. Eugene Schultz Dr. Gary Smith Dr. Eugene Spafford James T. Tippett Ken vanWyk Roy Wood Paul Woodie	National Security Agency National Computer Security Center Queensland University of Technology, Australia National Security Agency The Fortress Georgetown University National Institute of Standards and Technology National Institute of Standards and Technology General Research Northrup Grumman National Institute of Standards and Technology AGCS Harris Corporation Computers & Security National Computer Security Center Data Security Fidelity George Mason University Naval Research Laboratory National Security Agency Independent Consultant United States Secret Service Trusted Information Systems, Inc. Advanced Research Projects Agency Naval Research Laboratory Estee Lauder University of Pennsylvania Deloitte & Touche Stanford Research Institute, International Stanford Research Institute, International Trusted Information Systems, LTD Secure Solutions George Mason University Arca Systems, Inc. Stanford Research Institute, International Arca Systems, Inc. Coast Laboratory, Purdue University National Security Agency Defense Information Systems Agency National Security Agency National Security Agency
---	--

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By <i>per lti</i>	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	iii

Awards Ceremony

2:00 p.m. Thursday October 12
Convention Center, Room 310

The National Institute of Standards and Technology (NIST) and the National Computer Security Center (NCSC) will honor those vendors who have successfully developed products meeting the standards of the respective organizations. Immediately following the ceremony, honored vendors will have the opportunity to display these products.

The NCSC recognizes vendors who contribute to the availability of trusted products and thus expand the range of solutions from which customers may select to secure their data. The products are placed on the Evaluated Products List (EPL) following a successful evaluation against the Trusted Computer Systems Evaluation Criteria including its interpretations: Trusted Database Interpretation, Trusted Network Interpretation, and Trusted Subsystem Interpretation. Vendors who have completed the evaluation process will receive a formal certificate of completion from the Director, NCSC marking the addition to the EPL. Certificates will also be presented to those vendors that have placed a new release of a trusted product on the EPL by participation in the Ratings Maintenance Program. Additionally, vendors will receive honorable mention for being in the final stages of an evaluation as evidenced by transition into the Formal Evaluation phase. The success of the Trusted Product Evaluation Program is made possible by the commitment of the vendor community.

The Computer Security Division at NIST provides validation services to test vendor implementations for conformance to security standards. NIST currently maintains validation services for three Federal Information Processing Standards (FIPS): FIPS 46-2, Data Encryption Standard (DES); FIPS 113, Computer Data Authentication; and FIPS 171, Key Management Using ANSI X9.17. During this award ceremony, NIST presents "Certificate of Appreciation" awards to those vendors who have successfully validated their implementation of these standards.

With the reaffirmation of the Data Encryption Standard as FIPS 46-2 in 1993, DES can now be implemented in software, as well as hardware and firmware. To successfully validate an implementation for conformance to FIPS 46-2, a vendor must run the Monte Carlo test as described in NBS (NIST) Special Publication 500-20. The Monte Carlo test consists of performing eight million encryptions and four million decryptions, with two encryptions and one decryption making a single test.

Vendors test their implementations for conformance to FIPS 113 and its American National Standards Institute (ANSI) counterpart, ANSI X9.9, Financial Institution Message Authentication (Wholesale). This is done using an electronic bulletin board system. Interactive validation requirements are specified in NBS (NIST) Special Publication 500-156, Message Authentication Code (MAC) Validation System: Requirements and Procedures. The test suite is composed of a series of challenges and responses in which the vendor is requested to either compute or verify a MAC on given data using a specified key which was randomly generated.

Conformance to FIPS 171 is also tested using an interactive electronic bulletin board testing suite. FIPS 171 adopts ANSI X9.17, Financial Institution Key Management (Wholesale). ANSI X9.17 is a key management standard for DES-based applications. The tests are defined in a document entitled NIST Key Management Validation System Point-to-Point (PTP) Requirements. The test suite consists of a sequence of scenarios in which protocol messages are exchanged under specified conditions.

We congratulate all who have earned these awards.

18th National Information Systems Security Conference

Welcome Letter.....	<i>i</i>
Conference Committee & Referees	<i>ii</i>
Award Ceremony	<i>iv</i>
Table of Contents	<i>v</i>
Authors Cross Reference.....	<i>xv</i>

Refereed Papers

TECHNICAL CHALLENGES, TRACK A

Enforcement of Complex Security Policies with BEAC.....	1
I-Lung Kao, Randy Chow, <i>University of Florida</i>	
The Controlled Application Set Paradigm for Trusted Systems	11
Daniel F. Sterne, <i>Trusted Information Systems, Inc.</i> ; Glenn S. Benson, <i>European Computer-Industry Research Centre</i>	
Information Domains Metapolicy.....	27
Gene Hilborn, <i>Computer Sciences Corporation</i>	
Maintaining Secrecy and Integrity in Multilevel Databases: A Practical Approach.....	37
Sushil Jajodia, <i>George Mason University</i> ; Don Marks, <i>Department of Defense</i> ; Elisa Bertino, <i>Universita di Milano</i>	
TOP: A Practical Trusted ODBMS.....	50
Marvin Schaefer, <i>Arca Systems, Inc.</i> ; Valeria A. Lyons, Paul A. Martel, Antoun Kanawati, <i>ONTOS, Inc.</i>	
Great Unsolved Problems in Applied Computer Security.....	63
Mark G. Graff, <i>Sun Microsystems</i>	
Addressing INFOSEC Analysis Problems using Rule-Based Technology	73
Richard B. Neely, Ph.D., James W. Freeman, Ph.D., <i>CTA Incorporated</i>	
Identification of Subjects and Objects in a Trusted Extensible Client Server Architecture.....	83
Terry C. Vickers Benz, E. John Sebes, Homayoon Tajalli, <i>Trusted Information Systems, Inc.</i>	
The New Alliance: Gaining on Security Integrity Assurance.....	100
René H. Sanchez, <i>Rockwell Space Operations Company</i> Donald L. Evans, <i>UNISYS</i>	
An Unusual B3-Complaint Discretionary Access Control Policy	113
Jeremy Epstein, Gary Grossman, Albert Donaldson, <i>Cordant, Inc.</i>	
GENSER Message Multi-Level Secure Classifications and Categories	123
Mary Lou Hoffert, <i>NCPII Development Team, NCTAMS LANT and NCTS Washington</i>	
A Standard Audit Trail Format	136
Matt Bishop, <i>University of California, Davis</i>	
TCP/IP (Lack of) Security.....	146
Jesper M. Johansson, <i>University of Minnesota</i>	
AINT Misbehaving--A Taxonomy of Anti-Intrusion Techniques.....	163

Lawrence R. Halme, R. Kenneth Bauer, *Arca Systems, Inc.*

Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions	173
Mandy Chung, Nicholas Puketza, Ronald A. Olsson, Biswanath Mukherjee, <i>University of California, Davis</i>	
Maintaining Privacy in Electronic Transactions	184
Benjamin Cox, <i>Carnegie Mellon University</i>	
A Software Architecture to Support Misuse Intrusion Detection	194
Sandeep Kumar, Eugene H. Spafford, <i>The COAST Project, Purdue University</i>	

SOLUTIONS, TRACK B

Providing Accurate Data Labels to the Analyst - The Secure C ⁴ I Workstation	205
Ingrid Dampier, Christine Corbett, <i>TRW Integrated Engineering Division</i>	
Controlling Network Communication with Domain and Type Enforcement	211
David L. Sherman, Daniel F. Sterne, Lee Badger, Sandra L. Murphy, Kenneth M. Walker, Sheila A. Haghighat, <i>Trusted Information Systems, Inc.</i>	
Integrating COTS Applications on Compartmented Mode Workstations	221
Susan A. Heath, <i>The Boeing Company</i>	
Project WINMILL: Using a COTS Solution to Connect LANs of Different Compartments.....	228
Al Nessel, Curt Sawyer, <i>Defense Intelligence Agency</i>	
On Guards . . . En Garde.....	236
Lawrence M. Sudduth, <i>Secure Computing and Communications, Inc.</i>	
Securing Local Area and Metropolitan Area Networks: A Practical Approach	249
Prof. Vijay Varadharajan, <i>University of Western Sydney, Nepean, Australia</i>	
Using Network Traffic Analysis as a Security Tool	262
Peter Troxell, Curry Bartlett, Nicholas Gill, <i>Digital Equipment Corporation</i>	
SAGE: Approach to Rapid Development of Trusted Guard Applications.....	271
Karen Goertzel, <i>Wang Federal, Inc.</i>	
Experiences with Implementing Messaging Security in MSMail 3.2	281
James E. Zmuda, Russell Housley, <i>Spyrus</i>	
Can Computers and Epidemiology Get Along? Health Problems in Computers	291
Guillermo M. Mallén-Fullerton MS, <i>Universidad Nacional Autónoma de México</i> ; Dr. Florencia Vargas-Vorackova PhD, <i>Instituto Nacional de la Nutrición</i> ; Dr. Enrique Daltaubuit-Godas PhD, <i>Universidad Nacional Autónoma de México</i>	
Disaster Recovery Planning Case Study: The South African 1994 Election	300
Walter Cooke, CISSP, <i>W. J. Cooke and Associates Ltd.</i>	
VHA's Approach to Contingency Plan Development	308
Gail Belles, <i>Medical Information Security Service</i> , <i>National Center for Information, VA Medical Center</i>	

CRITERIA AND ASSURANCE, TRACK C

Functional Security Criteria for Distributed Systems	310
Janet Cugini, <i>National Institute of Standards and Technology</i> ; Rob Dobry, <i>National Security Agency</i> ; Virgil Gligor, <i>University of Maryland</i> ; Terry Mayfield, <i>Institute of Defense Analyses</i>	

A Perspective of Evaluation in the UK Versus the US	322
Alan Borrett, Member of <i>UK ITSEC Scheme</i>	
ECMA's Approach for IT Security Evaluations	335
Alexander Herrigel, <i>R3 Security Engineering AG, Switzerland</i> ;	
Roger French, <i>Digital Equipment Corporation</i> ;	
Haruki Tabuchi, <i>Fujitsu Ltd, Japan</i> ;	
<i>The European Computer Manufacturers Association</i>	
Rating Network Components	344
Gloria Serrao, <i>National Security Agency</i>	
Analysis Requirements for Low Assurance Evaluations.....	356
James L. Arnold Jr., <i>National Security Agency</i>	
Measuring Correctness and Effectiveness: A New Approach Using Process Evaluation.....	366
Klaus Keus, Klaus-Werner Schröder, <i>Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany</i>	
Reengineering the Certification and Accreditation Process: Security is Free.....	374
Sean G. Mahon, <i>Boeing Information Services</i>	

MANAGEMENT AND ADMINISTRATION, TRACK D

Critical Factors of Key Escrow Encryption Systems	384
Dorothy E. Denning, <i>Georgetown University</i>	
Evaluating the Strength of Ciphers	395
John C. Higgins, <i>Brigham Young University</i>	
Community Response to CMM-Based Security Engineering Process Improvement	404
Marcia W. Zior, <i>National Security Agency</i>	
Measuring Security: What Can We Learn from Other Fields?	414
Deborah J. Bodeau, <i>The MITRE Corporation</i>	
Security and Software Reuse	424
George W. Rogers, Jr., Jerry C. Crabb, <i>The Analysis Corporation</i>	
The Use of Generic Architectures in System Integration.....	431
Dan Gambel, <i>General Research Corporation</i> ;	
Judith Hemenway, <i>Northrop Grumman Data Systems and Services Division</i>	
An Open Trusted Enterprise Network Architecture	447
Gary Grossman, Jeremy Epstein, <i>Cordant, Inc.</i> ;	
Roger Schell, <i>Novell, Inc.</i>	
Component Architectures for Trusted Netware.....	455
Jeremy Epstein, Gary Grossman, <i>Cordant, Inc.</i>	
Roger Schell, <i>Novell, Inc.</i>	
Social Engineering: The Only Real Test of Information Systems Security Plans.....	464
Ira S. Winkler, <i>Science Applications International Corporation</i>	
Contingency Planning: What to Do when Bad Things Happen to Good Systems	470
Jay J. Kahn, Marshall D. Abrams, <i>The MITRE Corporation</i>	
What Every Information Systems Security Professional Should Know	
About Electronic Records Management	480
Julie Smith McEwen, <i>CISSP, IIT Research Institute</i>	

THE INTERNET AND BEYOND, TRACK E

Computer Forensics: An Approach to Evidence in Cyberspace.....	487
Special Agent Mark M. Pollitt, <i>Federal Bureau of Investigation</i>	
Software Piracy: Prevention, Detection, and Liability Avoidance.....	492
Melissa J. Shaw, <i>Batelle</i>	
Authorship Analysis: Identifying the Author of a Program.....	514
Ivan Krsul, Eugene H. Spafford, <i>The COAST Project, Purdue University</i>	
Emerging Law Regarding Computers, Communications, and Software.....	525
J. Stewart Bradish, <i>University of Maryland</i>	
Internet Sniffer Attacks.....	534
E. Eugene Schultz, Ph.D., <i>SRI International</i>	
Thomas A. Longstaff, Ph.D., <i>Carnegie Mellon University</i>	
Information Warfare: A Front Line Perspective.....	543
Lieutenant Mark D. Tibbs, <i>U.S. Air Force</i>	
Defending a Computer System using Autonomous Agents.....	549
Mark Crosbie, Eugene H. Spafford, <i>COAST Laboratory, Purdue University</i>	

Special Unrefereed Papers

The Table of Contents for the 1st through the 17th National Computer Security Conferences.....	559
Jack Holleran, <i>National Computer Security Center</i>	
Darlene Affeldt, <i>National Security Agency</i>	
A Retrospective on the Criteria Movement.....	582
Willis H. Ware, <i>Rand Corporation</i>	
Conference Report: 17th National Computer Security Conference.....	589
Dennis Gilbert, <i>National Institute of Standards and Technology</i>	

Panel Summaries

TECHNICAL CHALLENGES, TRACK A

INFOSEC Research and Technology, Facing the Challenge: Secure Network Technology for the 21st Century.....	601
Joe Moorcones, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
Tom Zmurko, <i>National Security Agency</i>	
Dave Muzzy, <i>National Security Agency</i>	
Bill Ruppert, <i>National Security Agency</i>	
Blaine Burnham, <i>National Security Agency</i>	
Security on the I-WAY (High Speed ATM Networks).....	602
Ken Rowe, Chair, <i>University of Illinois Urbana-Champaign</i>	
<i>Panelists</i>	
Kem Ahlers, <i>Caterpillar, Inc.</i>	
Jay Dombroski, <i>San Diego Supercomputing Center</i>	
Ian Foster, <i>Argonne National Laboratory</i>	
Judy Warren, <i>Cornell Theory Center</i>	

Secure Database Systems: Where are We?.....	605
John R. Campbell, Chair, <i>National Security Agency</i>	
<i>Viewpoints</i>	
Statement	607
Richard Allen, <i>Oracle Corporation</i>	
Secure Database Systems - Where Are We?.....	609
Dick O'Brien, <i>Secure Computing Corporation</i>	
Directions for Database Security.....	611
Thomas Winkler-Parenty, <i>Sybase Inc.</i>	
Database Security for DoD and Commerce --- New Challenges	613
Bob Hedges, <i>Informix Software Inc.</i>	
Security in Infinite Networks	617
Ruth Nelson, Chair, <i>Information System Security</i>	
<i>Viewpoints</i>	
Managing Insecurity in Infinite Networks.....	619
Ruth Nelson, <i>Information System Security</i>	
Security Policies for Infinite Networks.....	621
Hilary H. Hosmer, <i>Data Security, Inc.</i>	
The VIP Security Paradigm	626
Dave Bailey, <i>Galaxy Computer Services</i>	
Closing the Gaps: Network Behavior Assessment.....	628
Kim Claffy, <i>San Diego Super Computer Center</i>	
Security for Infinite Networks.....	630
Steven M. Bellovin, <i>AT&T Bell Laboratory</i>	
Cryptographic Application Program Interface	631
Amy Reiss, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
John Linn, Panelist, <i>Open Vision</i>	
Piers McMahon, <i>ICL Ltd.</i>	
Dr. Burton Kaliski, <i>RSA Labs</i>	
The Future of Formal Methods for Security.....	634
Peter G. Neumann, Chair, <i>SRI International</i>	
<i>Viewpoints</i>	
Formal Methods and NASA	635
Ricky W. Butler, <i>NASA Langley Research Center</i>	
Algorithmic Verification	636
Robert Kurshan, <i>AT&T Bell Laboratories</i>	
Formal Methods: Changing Directions.....	637
Bill Legato, <i>National Security Agency</i>	

SOLUTIONS, TRACK B

Building a MLS System: A Real Life Adventure.....	638
Stephen Kougores, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
Gloria Fitzergald, <i>National Security Agency</i>	
Devloyn Arnold, <i>National Security Agency</i>	
Daphne Willard, <i>National Security Agency</i>	
Cindy Hash, <i>National Security Agency</i>	

Information Systems Security Research Joint Technology Office (Secure Virtual Office).....	641
John C. Davis, Chair, <i>National Computer Security Center</i>	
<i>Panelists</i>	
Dr. Howard Frank, <i>Advanced Research Projects Agency</i>	
Gregory Giovanis, <i>Defense Information Systems Agency</i>	
Teresa Lunt, <i>Advanced Research Projects Agency</i>	
Robert Meushaw, <i>National Security Agency</i>	
Developing an Incident Handling Capability.....	643
Marianne Swanson, Chair, <i>National Institute of Standards and Technology</i>	
<i>Viewpoints</i>	
Mark Graff, <i>Sun Corporation</i>	
Sandy Sparks, <i>Department of Energy's Computer Incident Advisory Capability</i>	
Sharon Sandstrom, <i>GE Information Services</i>	
An Assurance Framework or Can Process Replace Evaluation?.....	644
R. Kenneth Heist, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
William J. Marshall, <i>National Security Agency</i>	
John J. Adams, <i>National Security Agency</i>	
Stephen M. LaFountain, <i>National Security Agency</i>	
Dallas L. Pearson, <i>National Security Agency</i>	
Network Rating Model.....	647
Olga Lambros, Chair, <i>National Security Agency</i>	
<i>Viewpoints</i>	
Network Rating Model - Overview	650
Emily D. Joyce, <i>National Security Agency</i>	
Capability Maturity Models and their Role in the Network Rating Model	652
Dr. Bruce George, <i>National Security Agency</i>	
Quantifying Computer Security -- The Air Force C4 Systems Security	
Posture Model and Associated Metrics.....	655
Joe Filer, <i>Trident Data Systems, Inc.</i>	
Metrics: Their Role in the Network Rating Model	656
Colin Bowers, <i>National Security Agency</i>	

CRITERIA AND ASSURANCE, TRACK C

The TMach Experiment - Phase I.....	659
Ellen Colvin Flahavin, Chair, <i>National Institute of Standards and Technology</i>	
<i>Viewpoints</i>	
Helmut Kurth, <i>LABG</i>	
Julian Straw, <i>Logica/SISL</i>	
Nigel Rogers, <i>CESG</i>	
Martha Branstad, <i>Trusted Information Systems, Inc.</i>	
Common Criteria Editorial Board	662
Lynne Ambuel, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
Stephen M. LaFountain, <i>National Security Agency</i>	
Eugene Troy, <i>National Institute of Standards and Technology</i>	
Aaron Cohen, <i>CSE (Canada)</i>	
Yvon Klein, <i>SCSSI (France)</i>	
Chris Ketley, <i>CESG (UK)</i>	
Ulrich van Essen, <i>GISA (Germany)</i>	

The New OMB Circular A-130, Appendix III	663
Barbara Guttman, Chair, <i>National Institute of Standards and Technology</i>	
<i>Panelists</i>	
Scott Charney, <i>Department of Justice</i>	
Ed Roback, <i>National Institute of Standards and Technology</i>	
Ed Springer, <i>Office of Management and Budget</i>	
Perspectives on Internet Security Evaluation and Assurance.....	664
Bruce Aldridge, Chair, <i>NIST</i>	
<i>Panelists</i>	
Karin Taylor, <i>Communications Security Establishment, Canada</i>	
Marcus Ranum, <i>Information Works</i>	
Marvin Schaefer, <i>ARCA Systems, Inc.</i>	
Ron Ross, <i>Institute of Defense Analyses</i>	
Trusted Products - How Are They Used?	665
Laura M. King, Chair, <i>National Security Agency</i>	
Trust Technology Assessment Program.....	666
Thomas Anderson, Chair, <i>National Security Agency</i>	
<i>Panelist</i>	
Ellen Colvin Flahavin, <i>National Institute of Standards and Technology</i>	
The Development of Generally-Accepted System Security Principles	667
Will Ozier, Chair, <i>ISSA GSSP Committee</i>	
<i>Panelists</i>	
Marianne Swanson, <i>National Institute of Standards and Technology</i>	
Kristen Noakes-Fry, <i>Noakes-Fry Associates</i>	
Hal Tipton, <i>HFT Associates</i>	
Nigel Hickson, <i>Department of Trade and Industry</i>	

MANAGEMENT AND ADMINISTRATION, TRACK D

Linking Information Systems Security and Continuous Process Improvement:	
A Win-Win Organizational Strategy.....	668
Dennis Gilbert, Chair, <i>National Institute of Standards and Technology</i>	
<i>Viewpoints</i>	
Richard Belville, <i>Richard Belville and Associates</i>	672
Chris Bythewood, <i>National Computer Security Center</i>	674
Richard Koenig, <i>(ISC)²</i>	675
Corey Schou, <i>Idaho State University</i>	677
Ralph Spencer Poore, <i>Coopers & Lybrand L.L.P.</i>	678
INFOSEC Security Market, A Small Business Perspective	679
James P. Litchko, Chair, <i>Trusted Information Systems, Inc.</i>	
<i>Panelists</i>	
Jean Wu, <i>Information Systems Management, Inc.</i>	
Teresa Acevedo, <i>A & N Associates</i>	
Loreto Remorca, <i>Secure Solutions, Inc.</i>	

Will Encryption Keep Out the Hackers?.....	681
Dorothy E. Denning, Chair, <i>Georgetown University</i>	
<i>Panelists</i>	
Michael R. Higgins, <i>DISA/CISS</i>	
Stephen T. Kent, <i>BBN Communications Corporation</i>	
Eugene Spafford, <i>The COAST Project, Purdue University</i>	
<i>Viewpoint</i>	
Will Encryption Keep Out the Hackers?	682
Steven M. Bellovin, <i>AT&T Bell Laboratories</i>	
Commercial World: Requirements vs. Solutions / Corporate Security Challenges.....	683
Dennis Huamán, Chair	
<i>Panelists</i>	
Richard Lee	
Brian O'Higgins	
Stanley Jarocki	
National Information Infrastructure Security Initiatives, Part I.....	685
Electronic Commerce, Electronic Messaging (E-Mail) and Information Security, Overview of Panel	
Thomas Burke, Co-Chair, <i>GSA</i>	
F. Deane Erwin, Co-Chair, <i>NII SIPMO</i>	
<i>Panelists</i>	
Tom Clarke, <i>Defense Information Systems Agency</i>	
G. Martin Wagner, <i>ECA-PMO</i>	
<i>Viewpoints</i>	
Governmentwide E-MAIL VISION	686
Jack Finley, <i>GSA</i>	
Federal Electronic Commerce Program	687
Security Infrastructure Program Management Office	690
National Information Infrastructure Security Initiatives, Part II.....	693
Stephen Walker, Chair, <i>Trusted Information Systems, Inc.</i>	
<i>Viewpoints</i>	
Richard Rothwell, <i>USPS Electronic Commerce Services</i>	694
Jim Bidzos, <i>RSA Data Security, Inc.</i>	695
Nick Piazzola, <i>National Security Agency</i>	695
Wynn Redden, <i>Communications Security Establishment, Canadian Government</i>	696
INFOSEC, Prepare to Meet the New Millennium!.....	697
Dr. Charles Abzug, Chair, <i>Institute for Computer and Information Sciences</i>	
<i>Panelists</i>	
Marshall D. Abrams, , <i>The MITRE Corporation</i>	
Kevin T. Deeley, <i>Federal Bureau of Investigation</i>	
Patricia Edfors, <i>Department of Justice</i>	
Lynn McNulty, <i>McNulty and Associates</i>	
Donn B. Parker, <i>SRI International</i>	
Dr. Marv Schaefer, <i>Arca Systems</i>	
<i>Viewpoint</i>	
Information Security Infrastructure for the New Millennium	699
Dr. Roger R. Schell, <i>Novell, Inc.</i>	

THE INTERNET AND BEYOND, TRACK E

Legal Hacking - What is Computer Crime on the Internet?	703
Christine Axsmith, Chair, <i>Orkand Corporation</i>	
<i>Panelists</i>	
Scott Charney, <i>Department of Justice</i> ,	
Barbara Fraser, CERT, <i>Carnegie Mellon University</i>	
Dr. Lance Hoffman, <i>George Washington University</i>	
Marc Rotenberg, <i>Electronic Privacy Information Center</i>	
Law Enforcement Panel on Computer Forensics.....	705
Special Agent Mark M. Pollitt, Chair, <i>Federal Bureau of Investigation</i>	
<i>Panelists</i>	
Special Agent Stephen D. McFall, <i>Federal Bureau of Investigation</i>	
Special Agent Howard Schmidt, <i>United States Air Force Office of Special Investigations</i>	
Duncan Monkhouse, <i>Royal Canadian Mounted Police</i>	
<i>Viewpoint</i>	
Department of Maryland State Police Computer Crimes Unit	706
Sergeant Barry E. Leese, <i>Maryland State Police</i>	
Internet Security: Current Threats and Practical Solutions	708
John Wack, Chair, <i>National Institute of Standards and Technology</i>	
<i>Viewpoints</i>	
Trends in Internet Attacks and Unauthorized Access.....	708
David Curry, <i>Purdue University</i>	
Business Needs and Concerns with Internet Firewalls.....	708
John Pescatore, <i>International Data Group</i>	
WWW Security: Current Problems and Solutions, Future Trends	709
Robert Bagwill, <i>National Institute of Standards and Technology</i>	
Network Attacks Analysis: Stopping the Cycle of Internet Security Attacks, Alerts, and Patches .	709
Dr. Matt Bishop, <i>University of California, Davis</i>	
<i>The Internet Series</i>	
Internet Security	710
Jon David, <i>The Fortress</i>	
<i>Viewpoints</i>	
Weaknesses and Vulnerabilities of the Internet.....	712
Padgett Peterson, <i>Martin Marietta</i>	
Internet Security Tools	716
Steven M. Bellovin, <i>AT&T Bell Laboratories</i>	
Network Security Tools: Implementations and Implications.....	718
Paul Ferguson, <i>U.S. Sprint</i>	
Publication of Vulnerabilities and Tools	727
Sarah Gordon, <i>Command Software Systems, Inc.</i>	
Information Warfare: Its Impact upon Information Security	728
Wayne Madsen, Chair, <i>Computer Sciences Corporation</i>	
<i>Panelists</i>	
Martin R. Hill, <i>Office of the Assistant Secretary of Defense, C³I/IW</i>	
David Banisar, <i>Electronic Privacy Information Center</i>	
John Stanton, <i>Technology Transfer Journal</i>	
<i>Viewpoints</i>	
Information Warfare: Its Origins and Challenges for Information Security	730
John Hamlet, <i>Deacon House</i>	

THE TUTORIAL TRACK, TRACK F

Tutorial Series on Trusted Systems and Operational Security.....	735
Dr. Gary Smith, <i>ARCA Systems, Inc.</i>	
<i>Presenters:</i>	
Karen Ferraiolo, Mike Weidner, Stan Wisseman, Jack Wool, <i>ARCA Systems, Inc.</i>	
R. Quane, A. Strameela, <i>National Cryptologic School</i>	
Dr. Harold Highland, <i>Computers & Security</i>	
Dr. John Campbell, <i>National Security Agency</i>	
Joel Sachs, <i>The Sachs Group</i>	
Internet 101: Introduction to the Insecurity of the Internet.....	737
Dr. Harold Highland, FICS, Chair, <i>Computers & Security</i>	
<i>Panelists</i>	
Dr. Jon David, <i>The Fortress</i>	
Dr. Bertil Fortrie, <i>Internet Security News</i>	
Sarah Gordon, <i>Command Software</i>	
Padgett Peterson, <i>Martin Marietta</i>	
A Brief Database Security Tutorial: Or the less than Civil War between Ease-Of-Use and Security, the Battle between Grant and Lee's Privilege, Roles and Rollbacks, MAC DAC and FACT, even Distribution and Replication Maybe.....	740
John R. Campbell, Chair, <i>National Security Agency</i>	
From Training Standards to Courseware: An INFOSEC Success Story	758
Dr. Vic Maconachy, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
Dr. Corey Schou, <i>Idaho State University</i>	
Dr. John Cordani, <i>Eastern Michigan University</i>	
Dr. Timothy Mucklow, <i>U.S. Air Force</i>	
Lt. Ken Loker, <i>U. S. Navy</i>	
Ron Mayfield, <i>General Services Administration</i>	
MISSI Series.....	759
Brooke Jenkins, Chair, <i>National Security Agency</i>	
<i>Panelists</i>	
M. Fleming, <i>National Security Agency</i>	
S. Saydjari <i>National Security Agency</i>	
Todd Inskeep <i>National Security Agency</i>	
Carol Friedhoffer <i>National Security Agency</i>	
Al Arsenaault <i>National Security Agency</i>	
A Tutorial: The Internet, World Wide Web, and Beyond.....	760
Jeff Harrison, Chair, <i>National Institute of Standards and Technology</i>	

18th National Information Systems Security Conference

Author Cross Reference List

Abrams, Marshall D.	470	David, Jon	710
Abzug Charlie	697	Davis, John C.	641
Ahlers, Kem	602	Denning, Dorothy E.	384, 681
Aldridge, Bruce	664	Dobry, Rob	310
Allen, Richard	607	Dombroski, Jay	602
Ambuel, Lynne	662	Donaldson, Albert	113
Anderson, Thomas	666	Epstein, Jeremy	113, 447, 455
Arnold, James L., Jr.	356	Erwin, F. Deane	685
Axsmith, Christine.....	703	Evans, Donald E.	100
Badger, Lee	211	Ferguson, Paul	718
Bagwill, Robert	709	Filer, Joe	655
Bailey, Dave	626	Finley, Jack	686
Bartlett, Curry	262	Flahavin, Ellen Colvin	659
Bauer, R. Kenneth	163	Foster, Ian	602
Belles, Gail	308	Freeman, James W.	73
Bellovin, Steven M.	630, 682, 716	French, Roger	335
Belville, Richard	672	Gambel, Dan	431
Benson, Glenn S.	11	George, Bruce	652
Benzel, Terry C. Vickers	83	Gilbert, Dennis	559, 668
Bertino, Elisa	37	Gill, Nicholas	262
Bidzos, Jim	695	Gligor, Virgil	310
Bishop, Matt	136, 709	Goertzel, Karen	271
Bodeau, Deborah J.	414	Gordon, Sarah	727
Borrett, Alan	322	Graff, Mark G.	63
Bowers, Colin	656	Grossman, Gary	113, 447, 455
Bradish, J. Stewart	525	Guttman, Barbara	663
Burke, Thomas	685	Haghighat, Sheila A.	211
Butler, Ricky W.	635	Halme, Lawrence R.	163
Bythewood, Chris	674	Hamlet, John	730
Campbell, John R.	605, 740	Harrison, Jeff	760
Chow, Randy	1	Heath, Susan A.	221
Chung, Mandy.....	173	Hedges, Bob	613
Claffy, Kim.....	628	Heist, R. Kenneth	644
Cooke, Walter.....	300	Hemenway, Judith	431
Corbett, Christine	205	Herrigel, Alexander	335
Cox, Benjamin	184	Higgins, John C.	395
Crabb, Jerry C.	424	Highland, Harold	737
Crosbie, Mark	549	Hilborn, Gene	27
Cugini, Janet	310	Hoffert, Mary Lou	123
Curry, David	708	Holleran, Jack	559
Daltabuit-Godas, Enrique	291	Hosmer, Hilary H.	621
Dampier, Ingrid	205	Housley, Russell	281

Huamán, Dennis	683	Pescatore, John	708
Jajodia, Sushil	37	Peterson, Padgett	712
Jenkins, Brooke	759	Piazzola, Nick	695
Johansson, Jesper M.	146	Pollitt, Mark M.	487, 705
Joyce, Emily D.	650	Poore, Ralph Spencer	678
Kahn, Jay J.	470	Puketza, Nicholas	173
Kaliski, Burton	631	Redden, Wynn	696
Kanawati, Antoun	50	Reiss, Amy	631
Kao, I-Lung	1	Rogers, George W., Jr.	424
Keus, Klaus	366	Rothwell, Richard	694
King, Laura M.	666	Rowe, Ken	602
Koenig, Richard	675	Sanchez, René H.	100
Kougoures, Stephen	638	Sawyer, Curt	228
Krsul, Ivan	514	Schaefer, Marvin	50
Kumar, Sandeep	194	Schell, Roger R.	447, 455, 669
Kurshan, Robert	636	Schou, Corey	677
Lambros, Olga	647	Schröder, Klaus-Werner	366
Leese, Barry E.	706	Schultz, E. Eugene	534
Legato, Bill	637	Sebes, E. John	83
Linn, John	631	Serrao, Gloria	344
Litchko, James P.	679	Shaw, Melissa J.	492
Longstaff, Thomas A.	534	Sherman, David L.	211
Lyons, Valeria A.	50	Smith, Gary	735
Maconachy, W. V.	758	Spafford, Eugene H.	194, 514, 549
Madsen, Wayne	728	Sterne, Daniel F.	11, 211
Mahon, Sean G.	374	Sudduth, Lawrence M.	236
Mallén-Fullerton, Guillermo M.	291	Swanson, Marianne	643
Marks, Don	37	Tabuchi, Haruki	335
Martel, Paul A.	50	Tajalli, Homayoon	83
Mayfield, Terry	310	Tibbs, Mark D.	543
McEwen, Julie Smith	480	Troxell, Peter	262
McMahon, Piers	631	Varadharajan, Vijay	249
Moorcones, Joe	601	Vargas-Vorackova, Florencia	291
Mukherjee, Biswanath	173	Wack, John	708
Murphy, Sandra L.	211	Walker, Kenneth M.	211
Neely, Richard B.	73	Walker, Stephen	693
Nelson, Ruth	617, 619	Ware, Willis H.	582
Nessel, Al	228	Warren, Judy	602
Neumann, Peter G.	634	Winkler, Ira S.	464
O'Brien, Dick	609	Winkler-Parenty, Thomas	611
Olsson, Ronald	173	Zior, Marcia W.	404
Ozier, Will	667	Zmuda, James E.	281

Enforcement of Complex Security Policies with *BEAC*

I-Lung Kao* and Randy Chow

Department of Computer and Information Sciences and Engineering
University of Florida
Gainesville, Florida 32611
{kao, chow}@cis.ufl.edu

Abstract

Many computer applications in the commercial world need complex security policies which are hardly enforced by the military multilevel security model because their enforcement must violate the basic properties of the mathematical structure that the model is based on. Nor can these policies be modeled by a discretionary security model like the HRU's access control matrix since the accessing characteristics of these applications demand some degree of mandatory control. This paper presents an effective access control model called BEAC to enforce these complex security policies. The power of this model is demonstrated by its capability of expressing a rich set of access patterns from subjects to objects in an elegant and uniform way. Moreover, frequently-desired multilevel exceptions are systematically categorized and it is shown many security policies required by computer applications in commercial sectors are actually examples of these multilevel exceptions. Then it is demonstrated that all these multilevel exceptions and other commercial security policies can be effectively enforced by an extension of the BEAC model.

1 Introduction

1.1 Security policies and access control models

From the view point of access authorization, all system entities in a computing environment can be classified either as active *subjects* or passive *objects*. An *access control model* specifies how *security attributes* can be assigned to the interacting subjects and objects, and how these attributes are used in

evaluating access permission according to some prescribed rules. Given an access control model, an user of the system can define his *security policies* which specify how accesses from subjects to objects are to be regulated. An access control model provide a mechanism to enforce security policies. It is usually desirable to enforce as many security policies as possible with one uniform access control model.

Access control models are usually divided into two categories: mandatory access control and discretionary access control [20]. Both are formulated to allow or deny particular access modes by subjects to objects. The two categories of models differ mainly in how access authorizations can be modified. With a mandatory model, authorization modifications can only be made by an organization's security authorities by changing the security attributes of subjects and objects. In a discretionary model, a subject may be given some degree of freedom to pass the whole or part of its access privileges for an object to another subject.

Most mandatory access control models are lattice-based models, in the sense that each subject and object is associated with a security class, and the set of all security classes forms a lattice. All the classes in a lattice are partially ordered by a dominance relation. A model's access control rules reflect the security goal of the model and ensure that a subject can only have some mode of access (read or write) to an object when the security class of the subject dominates or is dominated by that of the object. The most well-known mandatory lattice-based models are the Bell-LaPadula multilevel model [1] for data confidentiality and the Biba multilevel model [2] for data integrity. In addition to security classes (hierarchical *levels*), it is often necessary to incorporate the *need-to-know* rule in the model for many commercial and military applications. The need-to-know rule is usually im-

*currently with IBM, Austin, Texas

plemented by a non-hierarchical component for the security attributes of subjects and objects, usually called *categories*. The categories, representing the natural characteristics of subjects and objects, also form a lattice with set containment as a basis of the dominance relation.

A discretionary access control model basically enumerates all the subjects and objects in a system and regulates the access to an object based on the identity of a subject or the groups to which it belongs [20]. It can be best represented by the HRU's access control matrix [12] with a row for each subject and a column for each object. Each entry of the matrix describes what access rights each subject has for each object. In this model, no semantics of information in the objects are considered, thus the security sensitivity of an object cannot be expressed. For performance reasons, an access control matrix is implemented by either a row-based mechanism (capability lists) or a column-based mechanism (access control lists), and both have their own pros and cons [9].

1.2 Needs for a new model

Because of its flexibility and adaptability to the needs of the real world's applications, category has been implemented as a basic mechanism for access control in some security systems (e.g., [10]). However, even with categories, conventional multilevel security models still cannot adequately enforce some security requirements needed by many applications. The most visible examples are different exceptions of multilevel information flow such as transitivity, aggregation, and separation (of duty) exceptions [11, 17] which all violate the basic properties of lattice, but are definitely required by many practical applications. Other security requirements that multilevel security models cannot satisfy are easily found. To incorporate these security requirements, system administrators are often forced to resort to less graceful and complicated methods to satisfy each requirement individually (e.g., [16, 22]). Thus, the difficulty of maintaining a secure computing environment satisfying all specific security requirements is increased considerably. These security requirements cannot be enforced by a discretionary access control model either, since the accessing characteristics of these applications demand some degree of mandatory control. Therefore, there is a need for a uniform and simple security model for enforcing security policies where both mandatory multilevel security and discretionary security models are inadequate.

With the above reasoning, the paper proposes a powerful access control model based on boolean expressions of categories. The model can implement a very rich set of regulated access patterns from subjects to objects in a natural and elegant way. Furthermore, it is shown that this model has a greater modeling power than conventional multilevel security models. We also systematically categorize the multilevel information flow exceptions in terms of access control. The model is then extended to incorporate the concept of states which must be supported in order to enforce these exceptions, and it is demonstrated how these multilevel exceptions and other complex security policies can be enforced by using the extended model.

2 A Model Based on Boolean Expressions

2.1 The basic model

Like most access control models, the proposed model divides all the entities in a system into subjects and objects. The security attribute of each subject is a category set which generally specifies the accessing characteristics of a subject. Unlike those in multilevel security models, the categories used here do not need to form a lattice. A category can also be created and assigned to a subject to enforce a desired security policy. The security attribute of each object is a boolean expression of categories, which basically is composed of categories assembled by any operators allowed in boolean algebra ("*" means AND, "+" means OR, and a bar over a category, e.g. \bar{c} , means negation), and is called an *Access Control Expression* in this paper, abbreviated as *ACE*. When a subject tries to access an object, the access is granted if the *ACE* of the object is evaluated TRUE using the subject's category set. The evaluation process of an *ACE* is described as follows: Any category in the object's *ACE* has a default value of 0. If an category c in the *ACE* also appear in the category set of the accessing subject, c is converted to TRUE in the *ACE*. The *ACE* is then evaluated according to the normal evaluation procedure in boolean algebra, and results in either TRUE or FALSE.

To define the model in a more formal way, if the category set of a subject S is represented by $CAT(S) = \{A\}$ and the access control expression for accessing an object O in some mode M by $ACE(O)_M = \langle E \rangle$, then the exclusive access control rule of this model is stated as "the access of S

to O in mode M is granted if $E(A) = \text{TRUE}$, where $E(A)$ means evaluating E with A as the input, and is denied if $E(A) = \text{FALSE}$."

The rules defined above apply to any access mode, such as read, write, or execute, etc., and an ACE can be independently defined for each access mode of an object. Whereas multiple access modes (thus multiple ACE s) might be defined for an object, for the reason of simplicity we will assume only one ACE with each object (thus one access mode only or one ACE applied to all access modes) in following discussions unless stated otherwise.

For example, if the category set of a subject S_i is $\{a, b, c\}$ and the ACE of an object O_j is $\langle a * \bar{c} \rangle$, S_i is not allowed to access O_j since the category c in $CAT(S_i)$ makes $ACE(O_j)$ false ($a * \bar{c} = \text{TRUE} * \overline{\text{TRUE}} = \text{TRUE} * \text{FALSE} = \text{FALSE}$). However, S_i is allowed to access another object O_k whose ACE is $\langle b + d + e \rangle$ since the existence of b in $ACT(S_i)$ makes $ACE(O_k)$ TRUE.

Taking an example of the government, a subject S_1 which represents an employee in the Department of Defense could have a category set $\{North_Korea, Nuclear_Weapon\}$, which implies that S_1 has access privileges to the objects categorized as *North_Korea*, *Nuclear_Weapon*, or both. Another subject S_2 which represents an employee in the Department of States has a category set $\{North_Korea, China\}$, which implies that the responsibility of S_2 requires him to have access rights to the objects categorized as *North_Korea*, *China*, or both. Now if an object O_1 representing a secret document file has an $ACE = \langle North_Korea \rangle$, then it can be accessed by both S_1 and S_2 because *North_Korea* exists in both category sets of S_1 and S_2 . Another object whose $ACE = \langle Nuclear_Weapon \rangle$ can be accessed by S_2 (because the default value of *Nuclear_Weapon* is FALSE) but cannot by S_1 (because the *Nuclear_Weapon* in S_2 makes this ACE FALSE).

The "wildcard" character, represented by the symbol '\$', is also used in an ACE to represent any category except those already appearing in the ACE . Utilizing the wildcard character is very effective in achieving some desired access pattern precisely. For instance, an object whose $ACE = \langle a * b * \$ \rangle$ can be accessed only by a subject whose category set contains only a and b and nothing else. Note that the value of the wildcat character is always determined after the value-substitutions of all other categories in an ACE .

As a general rule for achieving desired access restrictions, the existence of a category "c" in an object's ACE implies that a subject needs to have a

"c" in its category set in order to access the object, and a " \bar{c} " in an object's ACE implies that the object can only be accessed by a subject which does not have a "c" in its category set. Moreover, two categories appearing as " $c_i * c_j$ " in an object's ACE indicates that a subject must have both " c_i " and " c_j " in its category set to access the object, and two categories appearing as " $c_i + c_j$ " in the ACE means that any subject which has either " c_i " or " c_j " can access the object.

For simplicity in description, this boolean expression based access control model is named *BEAC*.

2.2 Modeling power

The modeling capability of the *BEAC* model is quite powerful. Firstly, it offers a flexible and elegant mechanism of access control. Both authorized and prohibitive access control can be expressed explicitly at the same time by one mechanism. The use of boolean expressions is more natural to enforce the security requirements of some real applications, especially in commercial sectors, than using the set containment relation in multilevel security models. The wildcard category used to generalize access patterns sometimes or to restrict them at other times is as powerful as using the wildcard character "*" in UNIX shell commands. The desirability of prohibitive rights and wildcard in specifying access rights is debatable [6]. However, the flexibility these mechanisms provide is useful for some special purposes as shown in the following.

Figure 1 shows how a complete set of access control among subjects to an object can be provided by the use of boolean expressions. Assume a system consisting of three subjects S_1 , S_2 , and S_3 with $\{a\}$, $\{b\}$, and $\{a, b\}$, respectively, as their category sets (e.g., S_1 and S_2 are two different employees, and S_3 is their manager), and one object called O (e.g., a document). Because any subject is either allowed or denied access to O , the total number of all possible access patterns of these three subjects to O is eight. By specifying the ACE of O appropriately, it can be seen in the figure that any of these eight access patterns can be precisely enforced by the *BEAC* model.

For a comparison with multilevel security models, it has been shown [13] that the *BEAC* model is powerful enough to enforce all the security policies that multilevel security models with levels and categories can enforce. That is, all the security policies for accessing objects previously enforced by a multilevel model can be exactly preserved using the

Subject	Category Set
S_1	{ a }
S_2	{ b }
S_3	{ a, b }

ACE of O	S_1	S_2	S_3
$\langle a \rangle$	X		X
$\langle b \rangle$		X	X
$\langle a + b \rangle$	X	X	X
$\langle a * b \rangle$			X
$\langle \bar{a} \rangle$		X	
$\langle \bar{b} \rangle$	X		
$\langle \bar{a} + \bar{b} \rangle$	X	X	
$\langle \bar{a} * \bar{b} \rangle$			

Figure 1: Eight access patterns of 3 subjects. An "X" in the entry means that subject S_i can access object O with the corresponding ACE.

BEAC model, by appropriately converting the levels and categories of all entities used in the multilevel model to the categories sets and ACEs used in *BEAC*.

On the other hand, it is interesting to show that there exists some security policies that can be enforced by the *BEAC* model but cannot by the multilevel access control model with categories. Suppose a system contains two subjects, S_1 and S_2 , and two objects O_1 and O_2 , and a security policy is applied to them such that the allowable and disallowed accesses to objects by subjects are shown in Figure 2. Both subjects can write information to both objects, but only S_1 can read information from O_1 and only S_2 can read information from O_2 . An application which needs this policy is that S_1 acts as a processing filter for O_1 such that any information written to O_1 must be read and processed by S_1 before it can be written to other objects again. S_2 plays the same role to O_2 . Another application is that O_1 is the mailbox of S_1 and O_2 is the mailbox of S_2 . Any subject may send messages to any mailbox but only the owner of a mailbox may read information from it.

First we show how this security policy can be enforced by the *BEAC* model. S_1 and S_2 have category sets {a} and {b}, respectively, according to their natural characteristics. O_1 can be written by both S_1 and S_2 but can be read only by S_1 , thus O_1 's ACE for write access is $\langle a + b \rangle$ and its ACE for read access is $\langle a * \bar{b} \rangle$. O_2 can be written by both S_1 and S_2 but can be read only by S_2 , thus O_2 's ACE for write access is also $\langle a + b \rangle$ and its ACE for read access is $\langle \bar{a} * b \rangle$.

However, it is infeasible to model the same security policy in Figure 2 using the multilevel access control with categories. Since S_1 can both read and write O_1 , $class(S_1) = class(O_1)$. Sim-

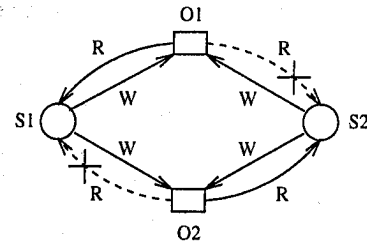


Figure 2: An access control policy which can be enforced by the *BEAC* model but cannot by the multilevel security model with categories.

ilarly, $class(S_2) = class(O_2)$. Moreover, since S_2 can only write but not read O_1 , the category set of O_1 must properly contain the category set of S_2 (if data confidentiality is the security concern), i.e., $class(O_1) \supset class(S_2)$, which implies $class(S_1) \supset class(S_2)$. However, with the same reasoning, the category set of O_2 must properly contain the category set of S_1 , i.e., $class(O_2) \supset class(S_1)$, which implies $class(S_2) \supset class(S_1)$ — a contradiction. Therefore, this security policy cannot be possibly enforced by the multilevel access control model with only categories.

As an observation from the example above, we can conclude that any security policy, that is represented by an information flow graph with cycles consisting read and write edges among more than two system entities (e.g., $O_1 \rightarrow S_1 \rightarrow O_2 \rightarrow S_2 \rightarrow O_1$ in Figure 2), cannot be enforced by a lattice-based access control model.

3 A Classification of State Dependent Security Policies

Complex access control policies are characterized by state-dependent security requirements. Authorization of access to objects by a subject depends on the subject's past access history and its interaction with other subjects and objects. For examples, a subject S is not allowed to access object O_1 if it has already read object O_2 , or subject S_1 or subject S_2 can write object O_3 , but they together can not write O_3 . We will categorize a class of state-dependent access control policies in terms of exceptions to normal information flow. Information flow is a different view from authorization control, but also need to be implemented by an access control model.

3.1 Multilevel information flow exceptions

An information flow model usually characterizes all system entities with different security classes and governs how information can flow between classes [15]. Traditional information flow models are built on a structure of lattice with components composing all the security classes, and information can only flow between components of the lattice in the direction as the properties used to construct the lattice permit [7, 8]. However, there exist some applications whose security requirements do need information flow which violates some properties of lattice. We will elaborate these *information flow exceptions* and use them as motivations for an extension of the *BEAC* model.

Information flow in a lattice-based model is transitive, i.e., if information is allowed to flow from class A to class B , and from B to class C , then it is allowed to flow from A to C directly. However, some applications exist where this transitive property is not desired. If we define the information flow relation " \rightarrow " on pairs of security classes to represent the allowable direction of flow and " \nrightarrow " to represent the prohibited direction of flow, then *transitivity exception* is formalized as $A \rightarrow B$ and $B \rightarrow C$, but $A \nrightarrow C$.

Another exception of information flow which may be desired by some applications is *aggregation exception* [17, 18]. In a lattice-based model, if $A \rightarrow C$ and $B \rightarrow C$, then the aggregate of information from A and B , represented as $A \cup B$, usually can flow to C . If this property is not desired, then we have an aggregation exception, which is formalized as $A \rightarrow C$ and $B \rightarrow C$, but $A \cup B \nrightarrow C$. This exception can be interpreted in two ways. One is that C can not sink information from the aggregate of A and B (e.g., information from A and B are combined and mixed by sharing a common *pipe* or *FIFO* with C), and the other is that after C sinks information from either A or B , it can not sink any information from the other class.

The dual problem of aggregation exception is the separation exception. Separation of duty is one of the most important ingredients in security policies and models concerning data integrity [3, 5, 14, 19, 21]. With respect to information flow, it can be described as that information cannot flow from a single class, either A or B , to another class C but only the aggregate of information from A and B can, which in practice can be interpreted as once information transfers from either A or B to C , the other must also transfer information to C . The in-

formation flowed to C from the first entity will not be valid or meaningful to C until information flow from the second entity happens. This requirement cannot be satisfied by a lattice-based information flow model alone, so we call it *separation exception*, formalized as $A \cup B \rightarrow C$, but $A \nrightarrow C$ and $B \nrightarrow C$.

These exceptions place more constraints on information flow among different classes than permitted by a lattice-based multilevel model. We will show in sections 4 and 5 that the *BEAC* model can be extended to enforce these exceptions, but first we formalize flow exceptions in terms of access control.

3.2 Refining flow exceptions in access control

Although the three exceptions mentioned above originate from information flow policies, they can be redefined in terms of access control. In access control, the main operations for information transfer between entities are read and write. So $A \rightarrow B$ means subject A writes information to object B or subject B reads information from object A . Furthermore, an access control model is usually chosen for either data confidentiality or data integrity purpose. Therefore these information flow exceptions are classified according to how subjects and objects interact and the security purpose in the scope of access control (as shown in Figure 3). The following details each exception and justifies its significance with possible applications.

Let's first look at what transitivity exception looks like in access control. Transitivity exception in information flow ($A \rightarrow B$ and $B \rightarrow C$, but $A \nrightarrow C$) can be described in access control as a relation among two subjects and two objects in two different ways. The first concerning with integrity (Figure 3 [i]) is that subject S_1 can write object O_1 , O_1 can be read by subject S_2 , and S_2 can write object O_2 , but S_1 can not write O_2 directly. This actually simulates the concept of "well-formed transaction" for the commercial integrity policy [5]. The other way which concerns confidentiality (Figure 3 [ii]) is that object O_1 can be read by subject S_1 , S_1 can write object O_2 , and O_2 can be read by subject S_2 , but O_1 can not be read by S_2 directly. An example of this exception is that raw data (O_1) can not be read by some user (S_2) directly without being converted to a specific format (O_2) by some formatting software (S_1).

Aggregation exception can also be redefined in terms of access control according to whether the security concern is data integrity or data confidentiality. If data integrity is the concern (Figure 3

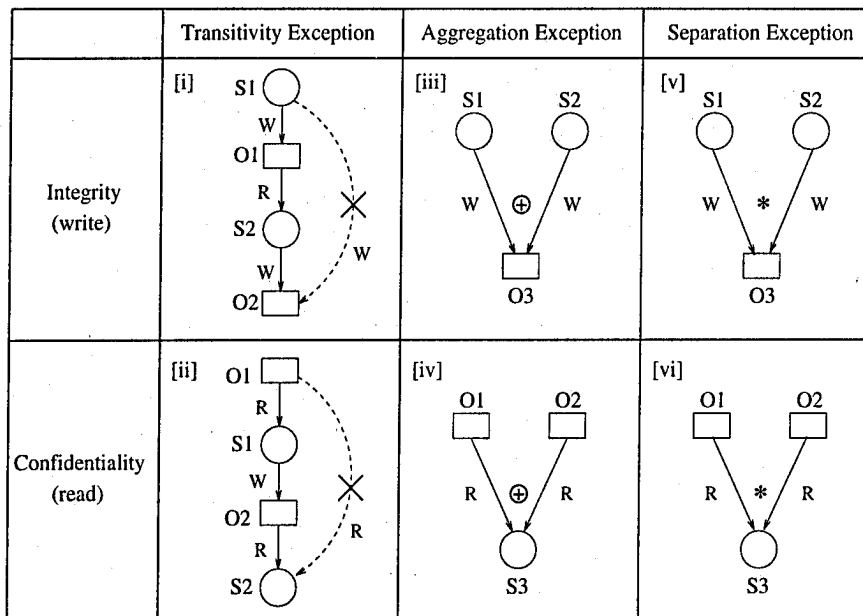


Figure 3: A taxonomy of information flow exceptions redefined in terms of access control. The meanings of symbols: “R”— read, “W” — write, “ \oplus ” — exclusive or, “*” — and.

[iii]), then either subject S_1 or subject S_2 can write object O_3 , but they together can not write O_3 . The interpretation is that after O_3 is written by S_1 , it cannot be written by S_2 any more, and vice versa. Any application which requires an object to be written by only one subject, but not a specific one, falls into this category of exception (e.g., an electronic check can only be prepared by only one accountant, and after it is prepared, no other accountants can touch it, to prevent against malicious modification). If data confidentiality is the concern (Figure 3 [iv]), then subject S_3 can read either object O_1 or object O_2 , but S_3 can not read the aggregate of both objects. This can be interpreted as that after S_3 reads O_1 , it can not read O_2 any more, and vice versa. A well-known example which generalizes this exception is the Chinese Wall security policy [4] in which a market analyst cannot access information from more than one company within the same interest class.

Since the original concern of separation exception is data integrity, many practical examples can be found in the literature discussing integrity policies and models (a simple one is that a check must be prepared and signed by two different accountants, to achieve separation of duty). It is described (Figure 3 [v]) as two subjects S_1 and S_2 accessing the object O_3 . After a subject (e.g., S_1) writes O_3 , only the other (S_2) is allowed to write that object. If data confidentiality is the concern (Figure 3 [vi]),

separation exception means that initially subject S_3 is allowed to read both objects O_1 and O_2 , but once after S_3 reads one object (e.g., O_1), it can only read the other object (O_2). An example similar to the one mentioned in [11] is that a user of a dial-up database may only read service charge information after he has viewed a stage of database information subscribed before he is allowed to view the next stage.

The *BEAC* model can be readily used to enforce the transitivity exceptions, by arranging categories sets of subjects and *ACEs* of objects appropriately [13]. However, to enforce aggregation and separation exceptions, the access privileges of a subject to an object needs to be affected either by the access of the other subject to the object or by the subject's earlier own access to other objects. It implies that some state information needs to be associated with subjects and objects such that the access privileges of subjects to objects will vary in different states. In the next section, we will extend the *BEAC* model to implement the state concept of the security attributes of subjects and objects.

4 The Extended *BEAC* Model

4.1 Analogy to the lock-key concept

BEAC has a great similarity with the lock-key concept used in discretionary access control [8]. The

lock-key concept is very intuitive in that a subject holding a key k_i which can be used to open a lock l_j can access the object "locked" by l_j . In the BEAC model, each category in an *CAT* virtually corresponds to a key, so the *CAT* of a subject corresponds to a set of different keys. On the other hand, the *ACE* of an object for one access mode corresponds to a "lock combination". An $ACE = \langle a * b \rangle$ represents a complex lock which can only be opened with presence of both keys a and b simultaneously. An $ACE = \langle a + b \rangle$ represents a generalized lock which can be opened by either key a or key b . An $ACE = \langle \bar{a} \rangle$ means a lock which remains open initially but the existence of key a in the *CAT* of a subject will lock it. More vividly, one *ACE* of an object represents a combination of locks on the door to the room where the object is located, and a subject must have all the necessary keys to open the door, in order to access the object in the access mode associated with that *ACE*.

4.2 Adding states by classifying categories

Motivated by the fact that access privileges of subjects to objects need to be restricted or expanded in order to enforce some complex security policies such as aggregation and separation exceptions, the security attributes of a subject and/or an object must be changed dynamically, as a result of access operations, yet in a controllable way. To facilitate this requirement, categories in the *CAT* of a subject are divided into two different classes. The first class is called *reusable category*, which permanently belongs to a subject once it is assigned to the subject, until a system security administrator explicitly removes it from the *CAT* of the subject through privileged commands. It is analogue to a reusable key which can be used by a subject to open a lock (an *ACE*) as many times as the subject would like to. The second class of categories is *one-time category*, which is dynamically assigned to a subject when the subject needs it. As its name implies, a one-time category can be used by a subject only once, and regardless whether it makes an *ACE* TRUE or FALSE, it is deleted from the *CAT* of the subject after its first use. (It can be imagined that a key is stuck on the door immediately after it is inserted into the lock hole, whether or not it can help to open the complex lock. A common mailbox in an apartment is one such example.) A category c is "used" only when a subject whose *CAT* contains c tries to access an object in a mode whose associated *ACE* also contains c . In other words,

a one-time category will not be removed from the *CAT* of an accessing subject if it does not appear in the *ACE* associated with that access mode. To differentiate these two classes of categories, a hat put on a category in a *CAT* is used to indicate a one-time category, e.g., \hat{c} .

The other way of changing a subject's privilege to an object by BEAC is to classify the categories composing the *ACE* of an object into two different classes. A *persistent category* is a category whose value remains TRUE once it is converted to TRUE. Contrasting with the lock-key concept, a persistent category corresponds to a lock which remains open once it is opened. A *non-persistent category* (lock), on the other hand, needs to be value-substituted (opened) each time the *ACE* is evaluated. Similarly, a \hat{c} in an *ACE* indicates that c is a persistent category.

It should be noticed that changing an object's security attribute has a greater effect than just changing a subject's security attribute, because the access privileges of all other related subjects will possibly be expanded or restricted. It should be used very carefully such that only the exact access control desired is achieved. To safeguard this, a more conservative approach is employed. It is assumed that whenever a new access control requirement is desired on an object, a new boolean expression is generated just for that requirement and is then ANDed with the original *ACE* (so the new generated boolean expression has no interference with the original *ACE*). To enforce a state-dependent complex security policy, both classifications of security attributes mentioned above are often required, as demonstrated subsequently.

5 Policy Enforcement with BEAC

5.1 Enforcing multilevel exceptions

In Section 3, multilevel information flow exceptions are categorized in terms of access control and justified by the security requirements of different applications. For brevity, only the enforcement of two exceptions by the BEAC model is demonstrated here. The other four exceptions can be similarly realized [13].

For clarity, all the security policies in this section use the conventions as follows:

- S_1, S_2, S_3, \dots : each represents a subject.
- O_1, O_2, O_3, \dots : each represents an object.
- $CAT(S_i)$: the category set of subject S_i .
- $ACE(O_j)_M$: the access control expression of ob-

ject O_j for access mode M .

- A, B, C, \dots : each represents a set of categories.
- p, q, r, \dots : each represents a reusable category in the category set of a subject or a non-persistent category in the ACE of an object.
- $\hat{p}, \hat{q}, \hat{r}, \dots$: each represents a one-time category in the category set of a subject or a persistent category in the ACE of an object.
- E, F, G, \dots : each represents a boolean expression.

aggregation exception - integrity

The original security attributes of subjects and objects are assumed to be:

$$\begin{aligned} CAT(S_1) &= \{A\}, \\ CAT(S_2) &= \{B\}, \\ ACE(O_3)_W &= \langle E \rangle, \end{aligned}$$

where A and B are two category sets which each makes E TRUE (note that A and B are not necessarily distinct). If we desire to enforce an aggregation exception between S_1 and S_2 to O_3 , we can change their security attributes as:

$$\begin{aligned} CAT(S_1) &= \{A, p\}, \\ CAT(S_2) &= \{B, q\}, \\ ACE(O_3)_W &= \langle E * (\bar{p} + \bar{q}) \rangle, \end{aligned}$$

where both p and q are newly created and do not exist in any of A, B, E . Since persistent categories \hat{p} and \hat{q} are complemented in the new ACE , they actually simulate a lock which is open to any subject unless the subject has both keys p and q (so changing the ACE of O_3 this way will not affect the access privileges of other subjects). Initially O_3 can be written by either S_1 or S_2 because a single p or q still can make the whole ACE TRUE. After S_1 , for example, writes O_3 , the value of \hat{p} in $ACE(O_3)_W$ will remain TRUE, which makes the ACE equivalent to $\langle E * \bar{q} \rangle$. When S_2 then tries to write O_3 , the ACE will be evaluated FALSE due to the category q in $CAT(S_2)$, so its access attempt will be denied.

separation exception - confidentiality

The original security attributes of subjects and objects are assumed to be:

$$\begin{aligned} ACE(O_1)_R &= \langle E \rangle, \\ ACE(O_2)_R &= \langle F \rangle, \\ CAT(S_3) &= \{A\}, \end{aligned}$$

where A is a category set which makes both E and F TRUE. If a separation exception is to be enforced between O_1 and O_2 for the read accesses by S_3 , their security attributes will be changed to:

$$\begin{aligned} ACE(O_1)_R &= \langle E * (p + \bar{r}) \rangle, \\ ACE(O_2)_R &= \langle F * (q + \bar{r}) \rangle, \\ CAT(S_3) &= \{A, \hat{p}, \hat{q}, r\}, \end{aligned}$$

where p, q and r are all new. The purpose of complementing r in the ACE s of O_1 and O_2 is not to

affect other subjects' privileges to these objects because of such an aggregation exception enforcement. Any other subject which originally has access to O_1 or O_2 still can access it since r does not exist in its category set. However, r is added to $CAT(S_3)$ so that the \bar{r} in either $ACE(O_1)_R$ or $ACE(O_2)_R$ does not open any door to S_3 . Initially S_3 can read either O_1 or O_2 . After S_3 read O_1 , for example, it will lose \hat{p} and make itself unable to read O_1 again since category p is non-persistent in $ACE(O_1)_R$. Therefore S_3 can then only be allowed to read O_2 .

5.2 Specifying a sequence of accesses

We now demonstrate another advantage of this model, i.e., its ability of assigning a fixed ordering to multiple subjects for their accesses to an object, in a straightforward way. For simplicity, the effect of modifying the ACE of an object upon access privileges of other unrelated subjects is not considered below. It can be eliminated, if necessary, by using the technique of adding a complemented category (\bar{r}) to the ACE s of objects and a non-complemented category (r) to the category sets of subjects involved in policy enforcement, as shown above.

Assume that three subjects S_1, S_2 , and S_3 can access an object O_4 in some mode M , so their security attributes are:

$$\begin{aligned} CAT(S_1) &= \{A\}, \\ CAT(S_2) &= \{B\}, \\ CAT(S_3) &= \{C\}, \\ ACE(O_4)_M &= \langle E \rangle, \end{aligned}$$

where A, B , and C all make E TRUE. If we desire to specify an access ordering to O_4 by three subjects as $S_1 \rightarrow S_2 \rightarrow S_3$, their security attributes are changed to:

$$\begin{aligned} CAT(S_1) &= \{A, \hat{p}\}, \\ CAT(S_2) &= \{B, \hat{q}\}, \\ CAT(S_3) &= \{C, \hat{r}\}, \\ ACE(O_4)_M &= \langle E * (p + \hat{p} * q + \hat{q} * r) \rangle, \end{aligned}$$

where categories p, q , and r are all new. It can be easily verified that at first only S_1 is allowed to access O_4 . After S_1 's access, the ACE of O_4 becomes $\langle E * (p + q + \hat{q} * r) \rangle$, which allows only S_2 to access O_4 . Then, after S_2 's access, the ACE of O_4 becomes $\langle E * (p + q + r) \rangle$, which only allows access to O_4 by S_3 .

The approach can be generalized to order accesses to an object by an arbitrary number of subjects.

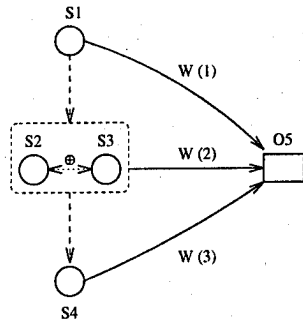


Figure 4: A complex security policy requiring both access ordering and aggregation exception for integrity.

5.3 Combination of enforcement techniques

Some complex security policies may require both exception and ordering. The following shows an example of the *BEAC* model using these techniques combined. Again, the effect of modifying the *ACE* of an object upon accesses of other unrelated subjects is not considered but could be eliminated using the technique mentioned earlier.

Assume there is a business application whose security requirement demands both access ordering and aggregation exception, as shown in Figure 4. An object O_5 (an electronic check) needs to be written by S_1 (a clerk) first, and then written by either S_2 or S_3 (two managers) but not both, and finally written by S_4 (another clerk). Assume their original security attributes are:

$$\begin{aligned} CAT(S_1) &= \{A\}, \\ CAT(S_2) &= \{B_1\}, \\ CAT(S_3) &= \{B_2\}, \\ CAT(S_4) &= \{C\}, \\ ACE(O_5)_W &= \langle E \rangle, \end{aligned}$$

where A , B_1 , B_2 , and C all make E TRUE. To enforce the security policy, we need to use the technique of specifying an ordering among S_1 , $[S_2 + S_3]$ (to treat them as one entity), and S_4 and the method of achieving aggregation exception for data integrity between S_2 and S_3 . Therefore, the security attributes become:

$$\begin{aligned} CAT(S_1) &= \{A, \hat{p}\}, \\ CAT(S_2) &= \{B_1, \hat{q}, \hat{s}\}, \\ CAT(S_3) &= \{B_2, \hat{q}, \hat{t}\}, \\ CAT(S_4) &= \{C, \hat{r}\}, \end{aligned}$$

$ACE(O_5)_M = \langle E * (p + \hat{p} * q * (\bar{s} + \bar{t}) + \hat{q} * r) \rangle$, where new categories p , q , r , s , and t do not appear in any of A , B_1 , B_2 , C , or E . Initially only S_1 can write O_5 , and after S_1 writes, $ACE(O_5)_M = \langle$

$E * (p + q * (\bar{s} + \bar{t}) + \hat{q} * r) \rangle$, which only allows either S_2 or S_3 to write O_5 . If S_2 writes, $ACE(O_5)_M = \langle E * (p + q * \bar{t} + r) \rangle$, then only S_4 can write O_5 .

6 Conclusions

Using the language of Boolean Algebra to achieve exact access patterns from subjects to objects is more precise and nature in meeting security requirements of many practical applications. The *BEAC* model proposed in this paper provides a systematic mechanism of modeling human-defined security policies by adequately assigning security attributes to both subjects and objects and using a simple access control rule to achieve the desired policy.

Furthermore, this model is extended from a stateless model to a more powerful version in which states are associated with subjects and objects simply by dividing their security attributes into two classes and render different meanings to different classes in access authorization. The overhead of implementing states on system entities by this way can be reduced to the minimum. As demonstrated in this paper, the modeling power of the extended model is surprisingly great. Many security requirements which cannot be adequately enforced by either conventional mandatory or discretionary security model, such as multilevel information flow exceptions, can be effectively enforced by the model.

While most multilevel security models assume only read and write operations on objects, the *BEAC* model does not specify any restriction on the set of access modes to an object and allows a single *ACE* for each access mode, thus providing a finer degree of access control. Independent control on each access mode is more flexible and desirable in current object-oriented systems, where a number of more abstract access operations can be defined on an object. Moreover, no predetermined security objective (confidentiality or integrity) is imposed in this model. Instead it just offers a practical mechanism for satisfying particular security policies. Information confidentiality or integrity may be achieved as just a property of the security policy to be enforced. This strategy is believed to be more consistent with the philosophy of separating policy and mechanism in the construction of modern security systems.

References

- [1] David E. Bell and Leonard J. LaPadula, "Computer Security Model: Unified Exposition

- and Multics Interpretation," Technical Report ESDTR-75-306, The MITRE Corporation, Bedford, MA, June 1975.
- [2] Kenneth J. Biba, "Integrity Considerations for Secure Computer Systems," Technical Report ESDTR-76-372, The MITRE Corporation, Bedford, MA, April 1977.
- [3] Lee Badger, "A Model for Specifying Multi-Granularity Integrity Policies," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1989, pp. 269 - 277.
- [4] David F. C. Brewer and Michael J. Nash, "The Chinese Wall Security Policy," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1989, pp. 206 - 214.
- [5] David D. Clark and David R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, April 1987, pp. 184 - 194.
- [6] F. Cuppens, "A Logical Analysis of Authorized and Prohibited Information Flows," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1993, pp. 100 - 109.
- [7] Dorothy E. Denning, "A Lattice Model of Secure Information Flow," *Communications of the ACM*, Vol 19, No. 5, May 1976, pp. 236 - 243.
- [8] Dorothy E. Denning, *Cryptography and Data Security*, Addison-Wesley, 1983.
- [9] Deborah D. Downs, et al., "Issues in Discretionary Access Control," *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, Oakland, CA, April 1985, pp. 208 - 218.
- [10] Todd Fine and Spencer E. Minear "Assuring Distributed Trusted Mach," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1993, pp. 206 - 218.
- [11] Simon N. Foley, "A Taxonomy for Information Flow Policies and Models," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1991, pp. 98 - 108.
- [12] Michael A. Harrison, et al., "Protection in Operating Systems," *Communications of the ACM*, Vol. 19, No. 8, August 1976, pp.461 - 471.
- [13] I-Lung Kao and Randy Chow, "Enforcing Complex Security Policies with Boolean Expression Based Access Control," *Technical Report UF-CIS-TR95-007*, University of Florida, February 1995.
- [14] Paul A. Karger, "Implementing Commercial Data Integrity with Secure Capabilities," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1988, pp. 130 - 139.
- [15] Carl E. Landwehr, "Formal Models for Computer Security," *Computing Surveys*, Vol.13, No. 3, September 1981, pp. 247 - 278.
- [16] Theodore M. P. Lee, "Using Mandatory Integrity to Enforce Commercial Security," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, April 1988, pp. 140 - 146.
- [17] Teresa F. Lunt, "Aggregation and Inference: Facts and Fallacies," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1989, pp. 102 - 109.
- [18] Catherine Meadows, "Extending the Brewer-Nash Model to a Multilevel Context," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1990, pp. 95 - 102.
- [19] Michael J. Nash and Keith R. Poland, "Some Conundrums Concerning Separation of Duty," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1990, pp. 201 - 207.
- [20] National Computer Security Center, "Department of Defense Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, December 1985.
- [21] Ravi Sandhu, "Transaction Control Expressions for Separation of Duties," *Proceedings of the 4th Aerospace Computer Security Application Conference*, 1988, pp. 282 - 286.
- [22] William R. Shockley, "Implementing The Clark/Wilson Integrity Policy Using Current Technology," *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988, pp. 29 - 37.

THE CONTROLLED APPLICATION SET PARADIGM FOR TRUSTED SYSTEMS *

Daniel F. Sterne
Trusted Information Systems, Inc.
3060 Washington Road
Glenwood, Maryland 21738
sterne@tis.com

Glenn S. Benson
European Computer-Industry Research Centre
Arabellastrasse 17
D-81925 Munich, Germany
benson@ecrc.de

Abstract

A fundamental assertion underlying the TCSEC paradigm is that all necessary automated security controls for a computer system can be provided by an operating system, in particular the components that constitute a conventional TCB. We challenge this assertion and explain why ordinary application processes outside an operating system can leak sensitive information, undermine an operating system's accountability mechanisms, and destroy information integrity.

We propose an alternative paradigm that more accurately identifies sources of security risk within a trusted system and can lead to improved security. The paradigm is based on the premise that every software component that can manipulate sensitive information, even if it has no special access control privileges, is potentially security relevant and must be controlled and protected by automated mechanisms. The paradigm repositions the trusted system security perimeter so that it encompasses not only an operating system TCB but the Controlled Application Set (CAS), a collection of components that have been screened and are presumed to be benign. The paradigm allows unscreened components to be present on a system but requires that they be prevented from manipulating sensitive information. A practical approach to assurance is outlined based on the notion of balanced assurance. Examples illustrate the applicability of the paradigm to systems providing confidentiality, accountability, and integrity.

*Funded by ARPA contract DABT63-92-C-0020 - Approved for Public Release - Distribution Unlimited.

1 Introduction

A fundamental assertion underlying the Trusted Computer Systems Evaluation Criteria (TCSEC) [10] is that all necessary automated security controls for many computer systems can be provided by their operating systems (OS), in particular the OS components that constitute a Trusted Computing Base (TCB). According to the TCSEC paradigm, if applications processes that have no special access control privileges are properly constrained by an OS TCB, they may safely execute software of unknown assurance while accessing sensitive information, i.e., information that merits special protection against unauthorized disclosure or modification. In particular, constraints based on a lattice are said to "confine" these untrusted subjects, thereby preventing them from causing security compromises. In this view, a robust application-independent TCB is like a "silver bullet" that protects sensitive information from errors and malicious code in the applications programs that manipulate it. In theory, if an OS TCB has been designed and implemented properly, the rest of the software in a system could be built by an adversary without undue risk of compromise.¹

The TCSEC conceptual architecture for a trusted system is shown in Figure 1. In the Figure, TCB components are shaded with a dark texture. The foundation of the TCB is the access control component or reference validation mechanism, shown as the bottom layer of the system. Other TCB components for identification and authentication (I&A), audit collection and storage, and other supporting functions are shown as a vertical column on the left. The TCB restricts access to both sensitive and non-sensitive information, represented by the cross-hatched and unshaded information storage containers below the sys-

¹Pottinger [21] attributes this assertion to Roger Schell.

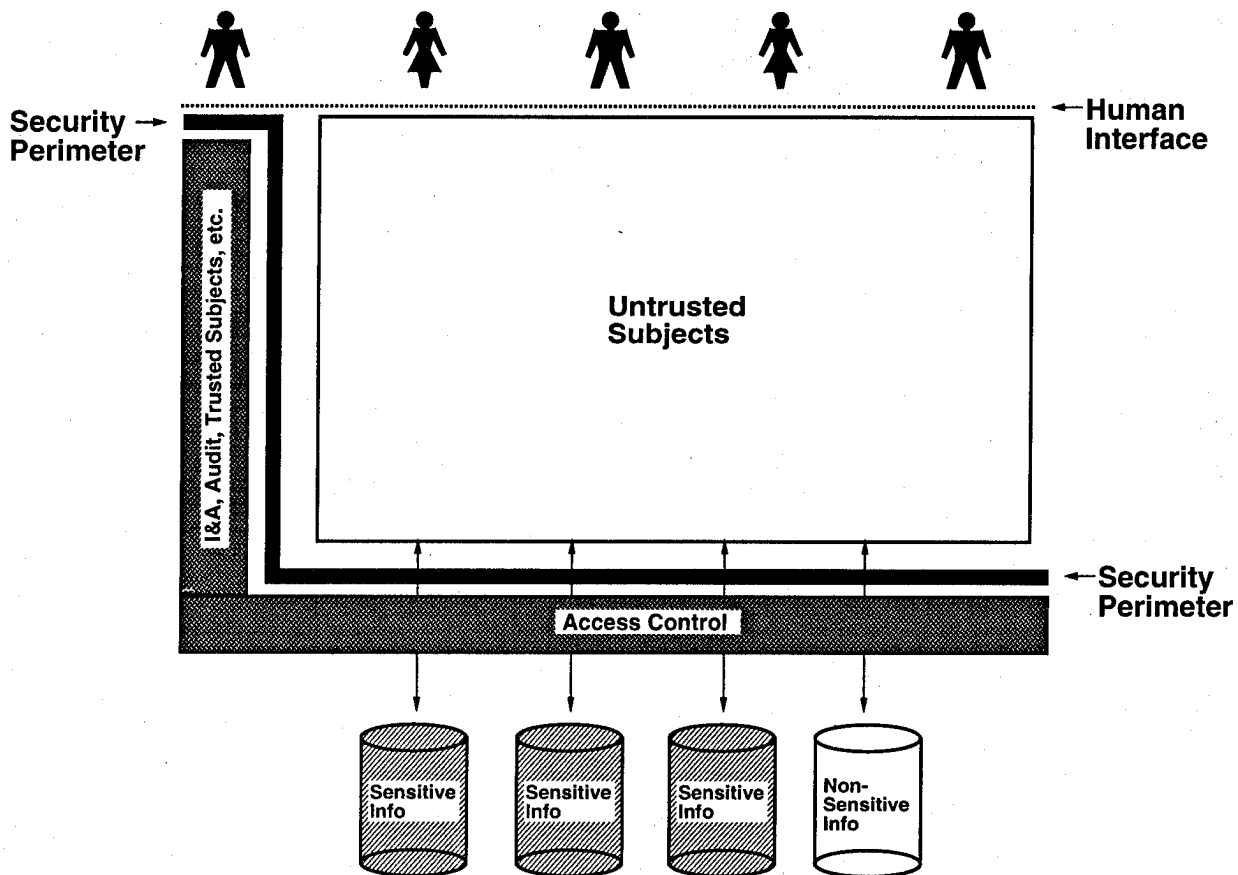


Figure 1: TCSEC Architecture and Security Perimeter

tem. Untrusted subjects, that is, subjects whose behavior is not security relevant, are shown atop the TCB's access control layer. (A component is security relevant if a system's ability to satisfy its security requirements depends on the component's behavior.) The human interface is shown above these subjects as a dotted line. As suggested by the arrows near the bottom, users direct untrusted subjects to manipulate both sensitive and non-sensitive information. Users may also interact directly with the TCB via the trusted path.

The TCSEC asserts that "the bounds of the TCB equate to the 'security perimeter'" (p. 67). The security perimeter is depicted in Figure 1 as a wide black border positioned between the TCB and untrusted subjects. Components below or to the left of the security perimeter are within the security perimeter and are security relevant; those above or to the right are outside the perimeter and are not security relevant. The portion of the security perimeter that converges

with the human interface in the upper left represents the trusted path.

A premise of this paper is that the confidentiality, accountability, and integrity protection needed by many organizations cannot be enforced unilaterally by application-independent components inside the perimeter. To provide the protection needed, the security perimeter must be repositioned outward, thereby acknowledging the security relevance of many ordinary application programs that the TCSEC paradigm treats as untrusted. This repositioning challenges the fundamental accuracy of the TCSEC paradigm as a guide for identifying and addressing sources of security risk within trusted systems. Such accuracy is critical because the TCSEC paradigm is the principal framework for conceptualizing, building, evaluating, and operating trusted systems.

This paper is organized as follows. Section 2 explains why the security perimeter must be repositioned if trusted systems are to provide better security. Sec-

tion 3 describes the Controlled Application Set (CAS) paradigm, comprising a proposed trust principle, a conceptual architecture, and an approach to assurance. Section 4 illustrates the applicability of the approach via examples dealing with confidentiality, accountability, and integrity. Sections 5 and 6 provide further discussion, including related work. Section 7 presents a summary and conclusion.

2 Why the Security Perimeter Must Be Repositioned

Under the TCSEC paradigm, TCBs allow untrusted software to manipulate sensitive information. As a consequence, even high-assurance TCBs fall short of meeting the computer security needs of many organizations in various ways, including the following:

- *Leakage*: Unless an OS TCB is completely free of covert storage and timing channels, it cannot by itself prevent sensitive information from being leaked to unauthorized users. Although paper designs for channel-free architectures based on exotic storage devices and other highly specialized techniques have been proposed in the research literature [22], *building* channel-free TCBs that are cost effective and provide acceptable system performance and functionality is beyond the state-of-the-art. Furthermore, there are no established techniques for systematically finding all covert channels in a TCB, let alone eliminating them. Moreover, as processor and I/O speeds increase, covert timing channel bandwidths will grow. Consequently, leakage vulnerabilities in TCBs are unlikely to diminish in the near term, if ever.
- *Accountability*: One of the control objectives and fundamental security requirements cited in the TCSEC is accountability, described as ensuring "that actions affecting security can be traced to the responsible party" [10]. Unfortunately, any program containing malicious logic can easily confuse the accountability mechanisms of an OS TCB. Suppose there are two subjects running such programs, each associated with a different user. Furthermore, suppose the subject running in the name of Smith forwards Smith's computational requests to the subject running in the name of Jones. If Jones's subject carries out the requests on Smith's behalf, the TCB's audit trail

will erroneously identify Jones as the "responsible party." To remedy this problem, the TCB could attempt to audit the forwarding of all such requests. This would require identifying all overt and covert means by which information can flow between different users' subjects, including subjects at the same security classification. This is a task at least as difficult as attempting to identify all covert downgrade channels in a multi-level secure (MLS) system. Worse yet, the TCB would have to *monitor the content* of all information exchanged between different users' subjects and distinguish illicit attempts to circumvent auditing from legitimate communication between users, clearly an impossible task.

- *Integrity*: An organization rarely defines the value of information solely in terms of confidentiality. To be useful to the organization, the information in addition must be accurate to some degree; information that is completely erroneous is of negligible value to an organization, even if rigorously protected from improper disclosure. Hence, nondisclosure requirements rarely exist apart from integrity requirements. An OS TCB by itself, however, cannot preserve the integrity of information because every program that a TCB allows to modify information is capable of corrupting it.

Although the TCSEC glossary defines the TCB as the "totality of mechanisms within a computer system ... responsible for enforcing a security policy," the examples above illustrate that an OS TCB cannot by itself provide the confidentiality, accountability, or integrity that many organizations need. Systems whose security relies on the TCSEC paradigm fall short because the security properties that are meaningful to system owners, such as leakage prevention and correct data transformations, cannot be enforced at the security perimeter depicted in Figure 1. Only by repositioning the perimeter outward so that it includes many additional application-dependent components can these properties be enforced.

3 The Controlled Application Set (CAS) Paradigm

The examples above suggest that meaningful system security requires cooperative interactions between an OS TCB and a collection of trustworthy applications. We use the term "application" broadly here

to mean any entity outside the TCB, including site-specific programs, operating system utilities, database management systems, and servers providing kernel-like services [11]. Based on this observation, we next propose an alternative paradigm for trusted systems. The key elements of the paradigm are a new trust principle, a conceptual architecture, and a practical approach to assurance.

3.1 Trust Principle

We propose the following as a general principle:

Any application that can manipulate sensitive information is potentially security relevant.

It follows from this principle that any application that can manipulate sensitive information must be controlled, requires some degree of assurance that it will exhibit only benign behavior, and must be protected from tampering.

We use the term *manipulate* as a shorthand to refer to access modes that are sensitive with respect to the security objective of interest. When confidentiality is the objective, assurance is needed for any application that can *read* sensitive information. When integrity is the objective, assurance is needed for any application that can *write* sensitive (high-integrity) information; under some circumstances, assurance may be unnecessary for applications that can read it. These distinctions are illustrated further in a later section.

Depending on an organization's security objectives and policies, *benign behavior* may mean, among other things, that an application will

- not exploit covert channels;
- not subvert accountability mechanisms, e.g., will refrain from performing services on behalf of one user in the name of another; or
- prevent certain kinds of information modifications identified *a priori* as harmful to integrity.

3.2 The CAS Conceptual Architecture

Figure 2 depicts the idealized CAS conceptual architecture, which repositions the trusted system security perimeter so that it encompasses not only an OS TCB but the Controlled Application Set (CAS), a collection of applications for which some assurance

of benign behavior has been obtained via an unspecified screening process.² TCB components, the human interface, and sensitive and non-sensitive information containers are shown as in Figure 1. CAS subjects, which are bound to CAS programs, are shown atop the TCB's access control layer. As suggested by the arrows in the Figure, users must use CAS subjects when manipulating sensitive information. In some systems, they may also be allowed to use CAS subjects to manipulate non-sensitive information, as shown.

The presence of the CAS forces a much larger part of the security perimeter to converge with the human interface. We envision this as widening the trusted path portion of the perimeter sideways from the left, rather than elevating the access control portion from the bottom. In this idealization, the CAS is not simply a layer on top of the TCB's access control component – it is *the layer*; it leaves no room for other non-CAS layers to be interposed between the user and sensitive information because such layers would be capable of causing the security problems noted earlier. In a later section, we discuss relaxing this constraint.

Other programs that have not been approved for inclusion in the CAS, including user-developed programs, may also reside on the system. However, any subject that executes a non-CAS program cannot be trusted. Untrusted subjects are shown in Figure 2 to the right of the CAS. To prevent such subjects from causing leaks and losses of accountability or integrity, we require that the TCB prevent them from manipulating sensitive information; they can manipulate only non-sensitive information, as shown. Such subjects are incapable of affecting the security of the system and can legitimately remain outside the security perimeter.

Untrusted subjects may be able to interact with CAS subjects in a constrained manner via sharable objects. In general, non-sensitive objects can be used for this purpose. Though not shown in the figure, sensitive objects can also be shared in some cases, as illustrated in Section 4.2.

Since the CAS is security relevant, CAS modifications and extensions must be carefully controlled. This has significant operational implications for systems in which users provide, develop, or enhance some of the programs they use. Consider a TCSEC-

²To minimize terminological confusion, we have chosen not to refer to the CAS as an element of a larger TCB. Using the term TCB to refer to all components inside the repositioned security perimeter, while technically correct, conflicts with common usage. Common usage, as exemplified by the National Computer Security Center's (NCSC) Evaluated Products List, is that the prototypical TCB is an application-independent OS.

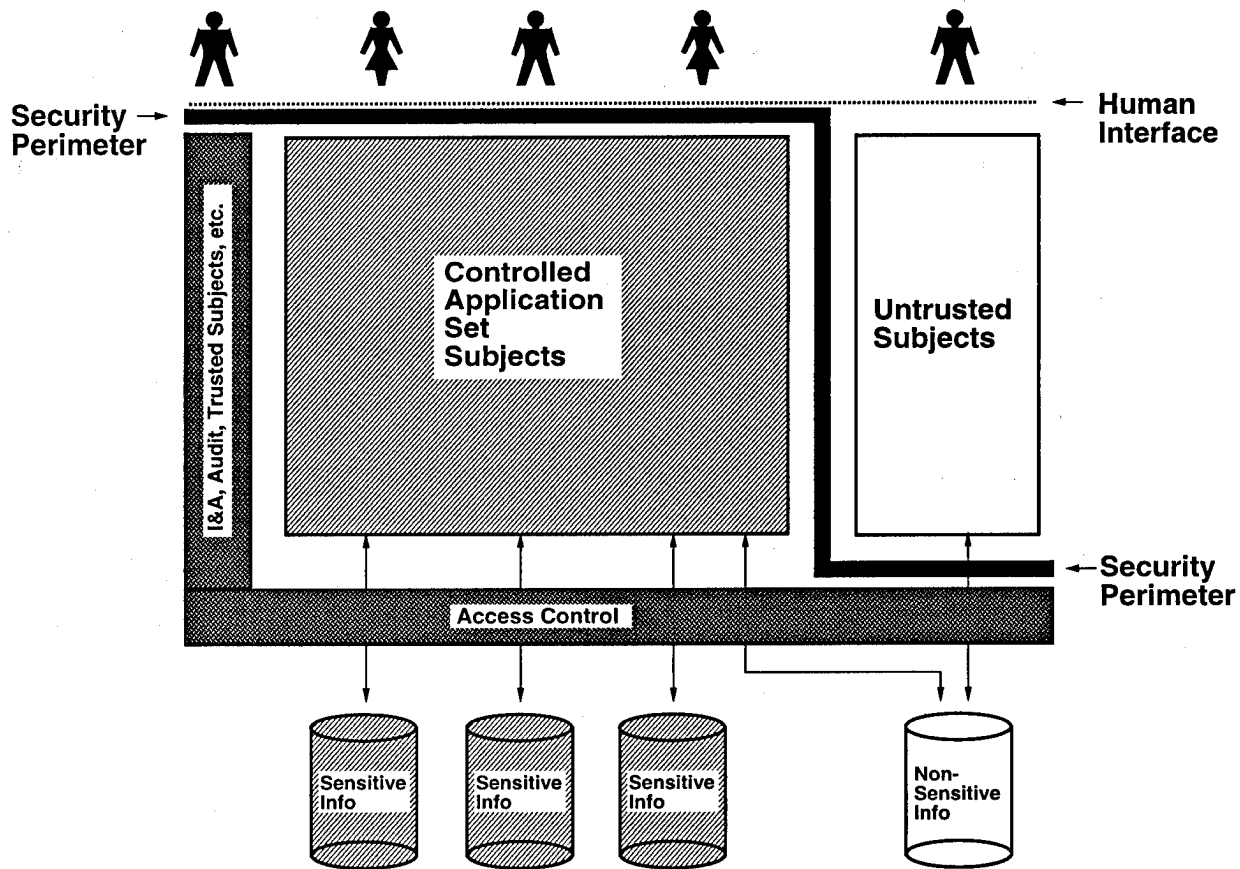


Figure 2: Controlled Application Set (CAS) Architecture and Security Perimeter

compliant MLS system serving both cleared and un-cleared users, where the former are accustomed to using their own programs to view, format, or edit classified information. In the CAS paradigm, cleared users would lose this ability. Unless installed in the CAS, user-developed programs will be able to manipulate only non-sensitive information. Techniques for lessening the potential operational burden associated with this restriction are discussed in Section 3.5.

3.3 The Role of the TCB

In the CAS paradigm, the TCB no longer comprises the "totality of protection mechanisms" responsible for security [10] because the totality now includes the CAS. Instead, the TCB acts as the *base* for these protection mechanisms, as implied by the phrase for which it stands: *Trusted Computing Base*. In this role, the TCB must extend to the CAS many of the facilities it uses to protect and support its own inter-

nal components. This constitutes a significant shift in its trust responsibilities.

The TCB by itself or in combination with particular components of the CAS must meet the following requirements:

Tamper Protection – The CAS must be protected from tampering. CAS subjects must run in a domain separate from those of non-CAS subjects. The CAS must not be modified without the explicit approval and participation of an authorized individual.

Non-Bypassability – The CAS must be non-bypassable. Every action that manipulates sensitive information must be accomplished via the CAS.

Trusted Path – As in the TCSEC, the TCB must support a trusted path between itself and users that can be invoked whenever a positive TCB-to-

user connection is required. In addition, the TCB must be able to transfer control from itself to the CAS at the request of a user so that a trusted path can be established between the CAS and the user and maintained continuously for the duration of any session in which sensitive information is manipulated.

Access to CAS Subjects and Programs –

CAS subjects must only be created on behalf of users who are authorized to manipulate sensitive information. Non-CAS subjects may execute CAS programs during non-sensitive sessions; these subjects, however, will not be granted any additional access rights to objects or the trusted path as a result.

Functionally Correct Services – The TCB must store, retrieve, and transform information in a manner that does not lessen the integrity of the information.

This last requirement stems from the use of the TCB as a base for other security mechanisms. If a TCB does not provide correct storage and retrieval services, no CAS component can be relied on to behave according to its security specification, source code, or documentation. A multiuser server process, for example, cannot be relied upon to provide user accountability if the TCB cannot store the server's audit logs correctly. Although the TCSEC imposes no requirements of this kind, they are *de facto* requirements for any useful operating system or security kernel.

For high-assurance systems, an additional requirement should be satisfied.

Multiple CAS Domains – The TCB must provide multiple execution domains for the CAS and restrict interactions among these domains as appropriate to the organization's security policy and assurance concerns.

This final requirement serves several purposes. First, it is a reinterpretation of the mandatory access control (MAC) requirements of the TCSEC in the following sense. Every subject controlled by a MAC-enforcing TCB runs in an execution domain implied by its MAC label [26]. Allowable interactions between the subjects operating in different MAC domains and objects are described by the read-down, write-up properties of the Bell-La Padula model [4]. The intended effect of restricting domain interactions in this way is that information cannot be transferred to less sensitive domains.

Second, it allows CAS domains to be arranged in different configurations to support other security policies, particularly policies concerned with integrity and role-based restrictions. For example, the configuration may form an "inverted" lattice [5] or may nest the domains so that one or more domains are subsets of others, thereby supporting the construction of CAS layers like TCB subsets [28]. Alternatively, the configuration may be nonuniform [6] in accordance with application-specific security policies [35].

Third, it supports the notion of least privilege [24] for the CAS. Since a CAS may be enormous, techniques for managing complexity are necessary if a CAS is to be of even modest assurance. An important technique is to organize the CAS as a collection of small tightly constrained domains in which CAS subjects are allowed to access only the objects essential to their assigned functions [17, 18]. This idea underlies a common interpretation of one of the TCSEC B3 requirements, namely, that a TCB's protection mechanism "shall play a central role in enforcing the internal structuring of the TCB."

3.4 A Practical Approach to Assurance

The obvious assurance issue confronting this approach is that a CAS may be extensive, encompassing millions of lines of software. As a consequence, developing CAS components according to the TCSEC and relying on the NCSC to evaluate them is infeasible. Fortunately, more pragmatic and modest assurance and evaluation practices will be entirely adequate in many cases.

3.4.1 Balancing Assurance and Risk

The CAS paradigm is based on the "balanced assurance" philosophy [15, 16], which asserts that the degree of assurance needed for a trusted component should be proportionate to the security risks the component poses. Since CAS components are protected and constrained by the TCB, CAS assurance risks can arguably be lower than those of the TCB; hence, less extensive assurance measures may be needed. This is particularly true for trusted systems in which the TCB enforces MAC constraints on the CAS. Accordingly, the level of assurance for many CAS components can and should be significantly *lower* than that of the TCB.

By contrast, the TCSEC paradigm treats CAS components as completely untrusted, requires no assurance for them whatsoever, and requires no support for them in the TCB. Hence, insisting on even minimal

CAS assurance and support cannot lessen the overall security of a trusted system and will in many cases significantly improve it.

Finer-grained assurance balancing may also be practical and beneficial. If information sensitivities or user authorization levels on a system vary greatly, CAS assurance requirements may need to vary on a component-by-component basis. An organization may deem that for some CAS components, or perhaps an entire CAS, very little assurance is required. On the other hand, for information that is extremely sensitive with respect to modification or disclosure, the organization may require that access to it occur only through a few extremely high-assurance CAS components.

3.4.2 Accountability of Origin

A common misconception in the TCSEC community is that one can trust an NCSC-evaluated TCB because evaluators have examined it thoroughly and forced the vendor to remove any security defects that might have originally been present. In fact, evaluators can only "spot check" a small fraction of a TCB's code and have little hope of finding such defects, particularly malicious code. Thompson [34] has pointed out that malicious code can be easily disguised from code inspectors and testers. As a result, even if evaluators could carefully inspect every line of code in an evaluated TCB, they still could not vouch for its purity with confidence. Inevitably, evaluators and customers have no choice but to trust that TCB vendors have not hidden malicious code in their products. They may be willing to trust them in this respect because they believe that vendors can be held accountable and that vendors have a vested interest in assuring the trustworthiness of their products.

In actuality, the NCSC evaluation process focuses on assessing that 1) the vendor is competent and employs suitable software development methods, and 2) the product meets minimum quality standards. A successful evaluation may increase confidence that a TCB will carry out certain functions correctly and uniformly (e.g., mediation) but cannot provide strong assurance that a TCB or other component is free of malicious code. Inevitably, one must trust the source of such components and can do so judiciously only if some organization or individual can be held accountable. In the CAS paradigm, accountability of origin is the most fundamental basis for trusting a CAS component; under no circumstances should a program for which there is no accountability (e.g., a program of unknown or highly questionable origin) be introduced into the CAS. The security benefits such re-

strictions provide are acknowledged in DoD's "Yellow Book" [33], which allows trusted systems to be used over a greater risk range if all applications are developed by cleared personnel.

3.4.3 Life Cycle Assurances

Beyond accountability of origin, an organization may require any of a broad range of life cycle and other assurance measures, including those cited in the TCSEC. To ensure that CAS sources are not only accountable but trustworthy and competent, an organization may impose personnel security (screening) or training requirements on CAS developers. Alternatively, it may be satisfied to obtain CAS components from certain reputable vendors. Quality control techniques may range from very stringent formal processes to highly informal procedures. Formal processes may involve independent verification and validation (IV&V), certification, or other forms of third-party oversight. They may additionally require construction of mathematical models, structured design reviews, extensive preoperation field testing, formal configuration management, trusted distribution, or trustworthy development environments. An informal process might simply require that a competent user vouch for each component installed in the CAS.

3.5 Supporting a Site-Extensible CAS

For a very high-assurance CAS, e.g., a CAS used to control a nuclear reactor or protect the information assets of a large financial institution, CAS change control procedures may be very restrictive. For a lower-assurance CAS, the owning organization may allow some of its members to modify or extend the CAS.

To improve system usability for these cases, a TCB together with the CAS should provide an "install" function that allows authorized users to promote new programs into specified CAS domains while the system is in operation. This function must only be available through the trusted path facility so that it can only be invoked with the explicit approval of a human being. It must not be possible for a program to invoke it automatically and invisibly. Moreover, its use must be auditable, so that installation of faulty components can be traced to the responsible party. Accountability of CAS changes can be further enhanced if necessary by other techniques, including use of one-time password authenticators. Organizations may choose to disable the install function altogether or selectively for particular CAS domains.

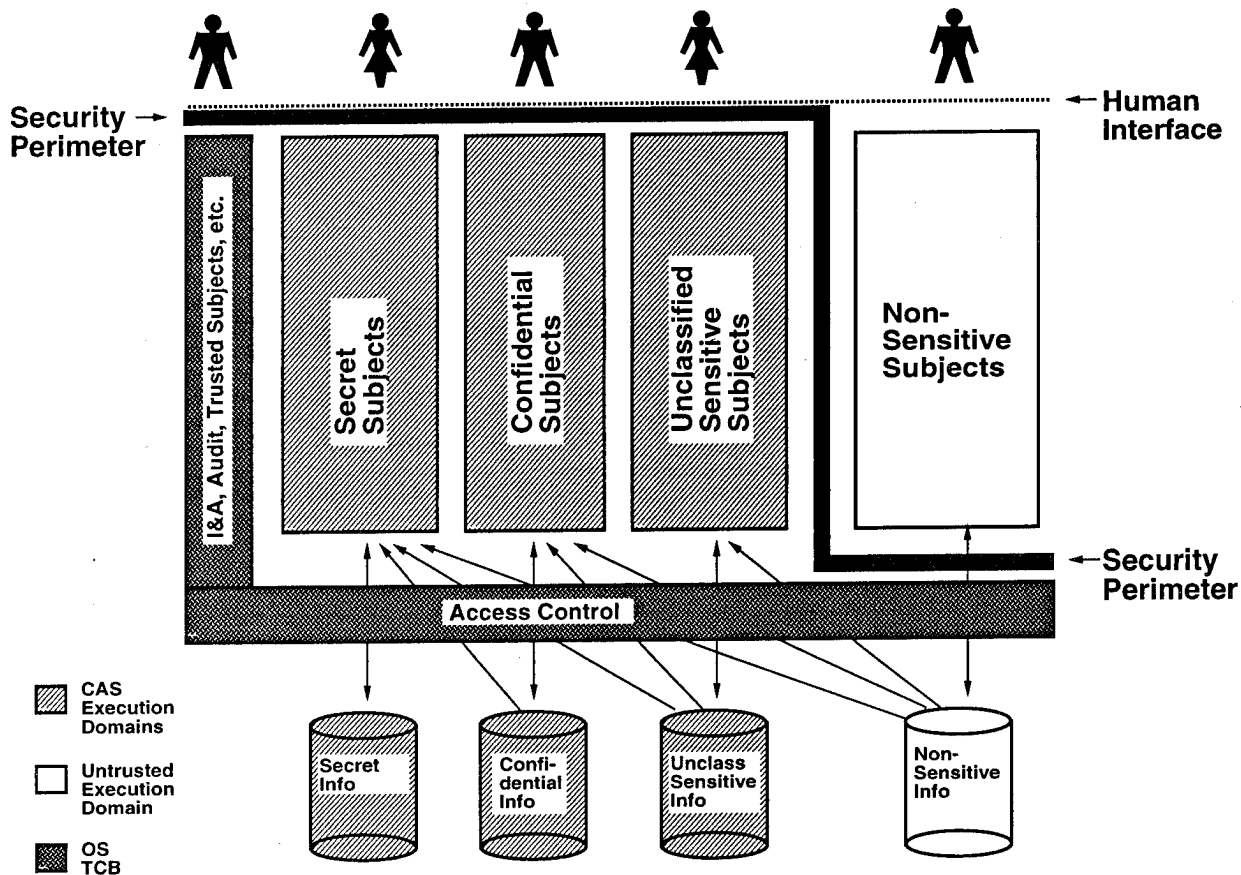


Figure 3: CAS Example - Confidentiality and Accountability

4 Examples

We now outline the way the CAS paradigm can be applied to a variety of systems.

4.1 Confidentiality and Accountability

This section discusses a hypothetical system trusted to enforce aspects of the U.S. laws, rules, and practices governing the protection of classified information. For brevity, we will focus on the mandatory rather than discretionary aspects of this policy.

In this example, the responsibilities of the trusted system include preventing electronic leakage of classified information to individuals who are authorized to use the system but are not sufficiently cleared. The system is also responsible for providing an audit log that lists the names of the users who have attempted to access or create classified information objects. As described above, an OS TCB by itself cannot address

these confidentiality and accountability requirements.

The CAS architecture for this system is shown in Figure 3. The system is based on an OS TCB that satisfies at least the B2 evaluation class requirements. The TCB has been extended to support a CAS and in particular provides multiple CAS execution domains. The system processes information of four different sensitivity levels: secret, confidential, unclassified sensitive, and non-sensitive. CAS subjects execute in the execution domains shown in the Figure as three cross-hatched rectangles labeled secret subjects, confidential subjects, and unclassified sensitive subjects. As suggested by the directional arrows near the bottom, these subjects are constrained by MAC. Nevertheless, they are trusted to not exploit the TCB's covert channels and not confuse the TCB's accountability mechanisms.

Non-CAS subjects execute in a domain in which only non-sensitive information can be accessed; they are represented in Figure 3 by the unshaded rectangle

to the right of the CAS, outside the security perimeter. Because they cannot access sensitive information, these subjects are incapable of leaking it or obscuring the identities of individuals who attempt to access it.

Among the users of this system are uncleared individuals. To mitigate the risk that a malicious program may leak secret information to an uncleared user, the organization that owns this system has imposed restrictions on the set of programs that can be installed in the CAS secret domain. Only commercial off-the-shelf (COTS) packages from approved vendors may be installed in this domain and only by a system administrator after approval by a configuration control board (CCB). Approval of a vendor may be based on the vendor's reputation, history, ownership, personnel security and software development practices, or other factors. User-developed programs may also be installed in the CAS secret domain by an administrator but only if supplied by a secret-cleared user and only after a CCB review of the source code. These restrictions are modest, yet they prevent users in secret sessions from inadvertently executing hostile programs planted by uncleared users. This significantly reduces the risks associated with covert channels in the TCB.

The organization allows users authorized for access to confidential and unclassified sensitive information to install programs in the corresponding CAS domains without participation of a system administrator. The primary security requirement for programs installed in the unclassified sensitive domain is that they not undermine the TCB's accountability mechanisms. Installing a program into either of these two domains is an auditable event and causes a copy of the program to be archived. These CAS mechanisms and procedures have negligible impact on users yet provide accountability protection that a TCB alone cannot.

There are no restrictions or special procedures associated with programs that execute in the non-sensitive domain.

4.2 Integrity and Accountability

In this section, we apply the CAS paradigm to a system trusted to enforce aspects of an integrity-oriented security policy [29] like that described by the Clark-Wilson integrity model [7]. These aspects are: 1) preventing unauthorized individuals from modifying sensitive information, 2) preventing authorized individuals from modifying such information in an unauthorized manner, and 3) recording in an audit log selected details about information modifications, e.g., user identifiers and the dollar amounts used in financial transactions. The system is shown in Figure 4.

Although this system's components perform different functions from those in the previous example, the essential security architecture is identical. As in the previous example, the TCB provides multiple domains for a CAS, but here the domain enforcement mechanism is programmable and supports a variety of domain configurations [6, 30]. The system protects the integrity of three kinds of sensitive information: salary and leave tables used by a payroll application, manufacturing specifications in the form of Computer Aided Design (CAD) drawings, and purchase orders. Authorization to modify these kinds of information is based on role assignment (job title) rather than clearance. The organization's policies and procedures state that only payroll clerks, senior engineers, and purchase officers, respectively, are authorized to modify these kinds of information and only via designated programs.

The system enforces these restrictions by associating a different CAS execution domain with each role, restricting the set of programs that can be executed in each domain, and allowing individuals to create subjects only in the domains for which they are authorized. The CAS programs that run in these domains are constrained by the TCB and can only modify the types of information appropriate for their associated roles. For example, programs that run in the Payroll Clerk domain can modify salary and leave tables but not CAD drawings. Each CAS program is trusted, however, to preserve information integrity by constraining the kinds of modifications that can be made, particularly to prevent fraud. For example, the payroll program prevents payroll clerks from modifying their own salaries or entering salaries above specific numerical thresholds.

Various kinds of uncontrolled information having no security relevance to the organization may also be kept on the system. The organization places no restrictions on the origin or behavior of programs used to modify them. The domain in which non-sensitive information alone can be modified is shown in Figure 4 as an unshaded rectangle outside the security perimeter. Since the security objective here is integrity, the CAS paradigm permits subjects outside the security perimeter to be given read-only access to sensitive information selectively, as depicted by the directional arrows in Figure 4. Information flow restrictions on interactions between subjects in different CAS domains may also be appropriate. Those shown here are illustrative only; where integrity is concerned, each organization must impose its own restrictions based on application-specific policies and assurance concerns.

The salary and leave tables on this system are an

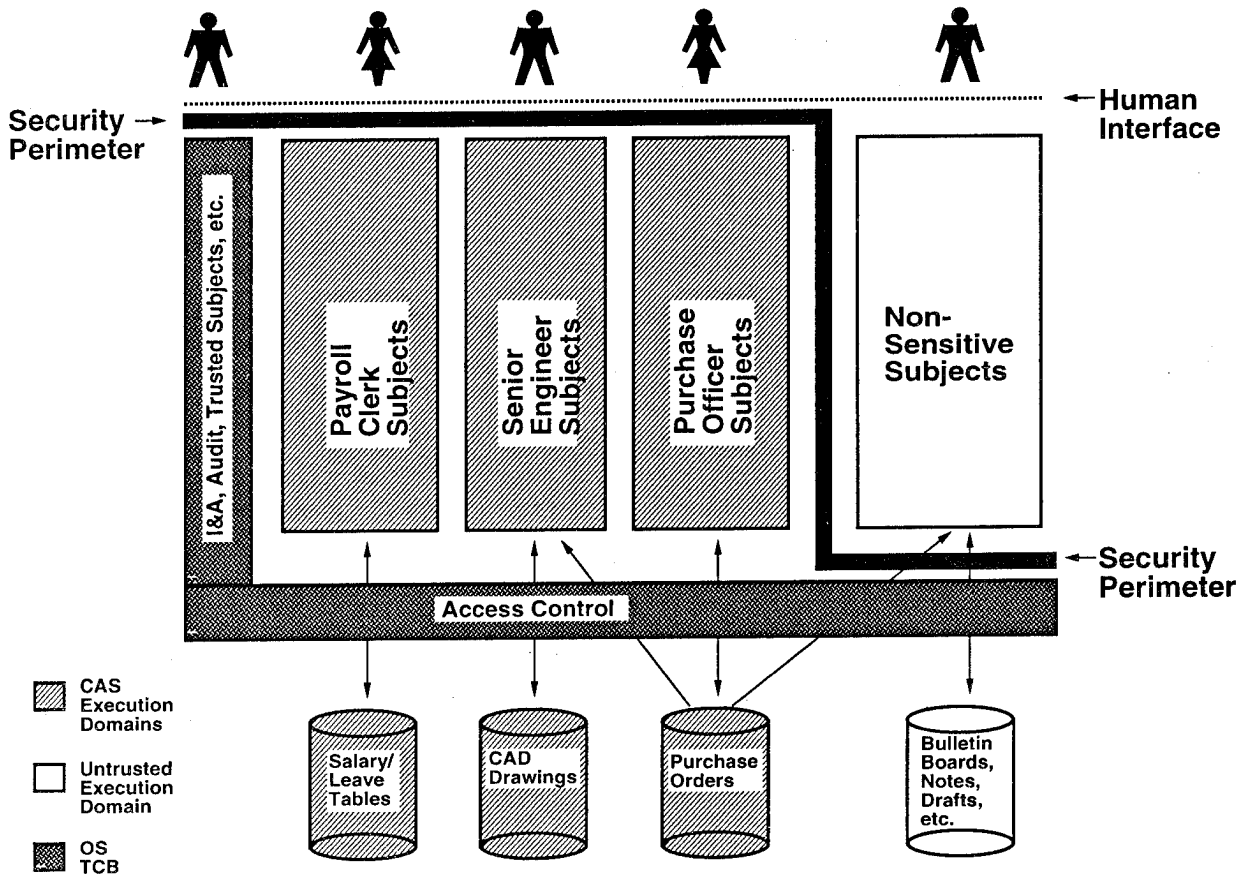


Figure 4: CAS Example - Integrity and Accountability

attractive target for electronic fraud. By preventing unauthorized individuals from modifying these tables, the TCB greatly reduces electronic fraud risks. Nevertheless, the organization considers it critical that the programs used to modify these tables be of very high assurance. Consequently, it has configured the system so that only a system administrator can install programs into the payroll clerk domain. The administrator is authorized to do this only after approval by the CCB. Assurance requirements for programs in other CAS domains are less stringent; these programs can be installed or revised more easily.

4.3 Low Assurance Systems

The previous examples illustrated the applicability of the CAS paradigm to multiuser systems built on high-assurance TCBs. This section applies the paradigm to a personal computer (PC) system equipped with a low-assurance TCB that provides no

features beyond those minimally required to support a CAS. This example reduces the CAS paradigm to its essence and reveals the most fundamental responsibilities of a TCB.

The security architecture for the system is shown in Figure 5. The security objective is a form of integrity, namely protecting tax returns, home finances databases, term papers, and other sensitive information from modification or deletion by computer viruses. A variety of non-sensitive information is also stored on the PC. This information is useful but does not merit special protection, e.g., copies of postings from network news groups.

The CAS consists of software that the PC owner has decided to trust to be free of viruses, including shrink-wrapped products from certain vendors and programs that have been scanned for viruses or digitally signed by their authors [23]. The owner would also like to run other (non-CAS) software without having to trust it in this manner. Non-CAS software includes free-

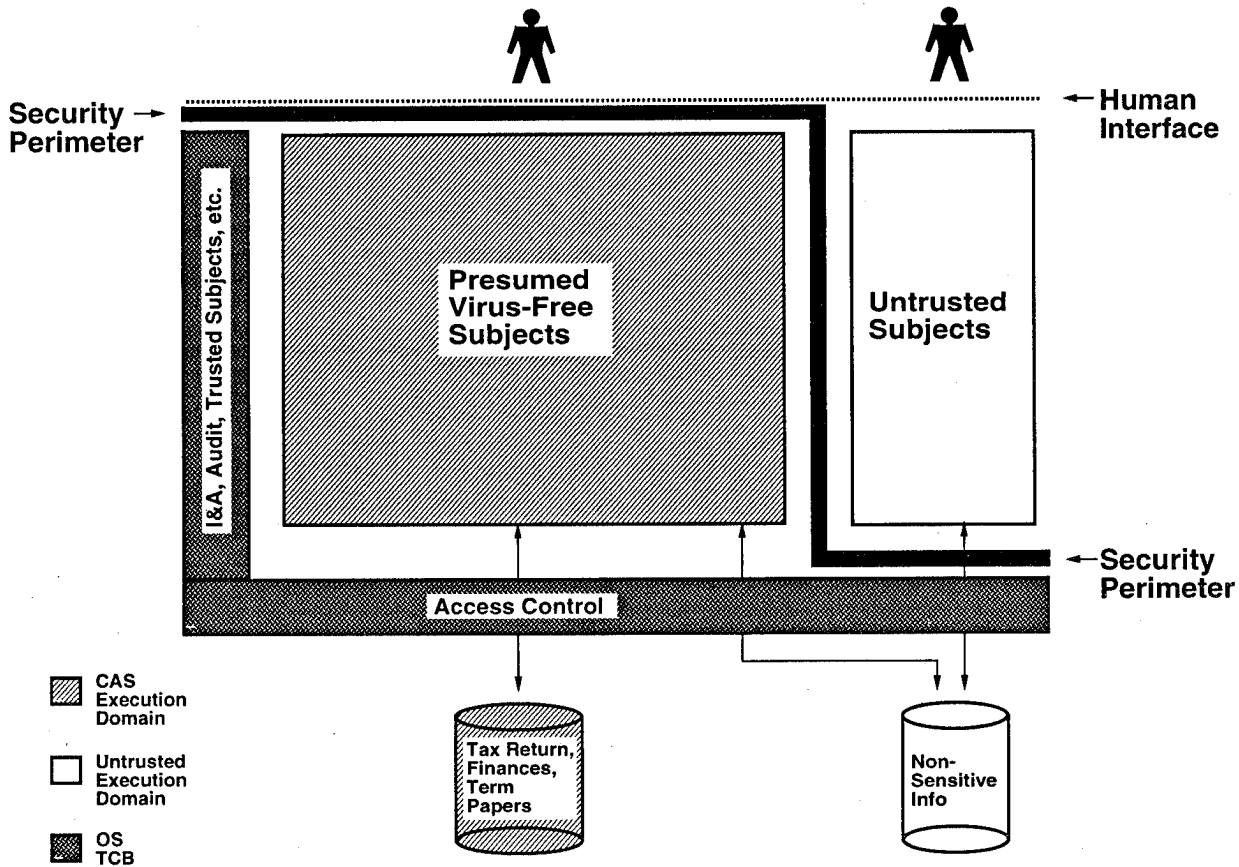


Figure 5: CAS Example - Personal Computer

ware of unknown origin and complex network applications that can automatically download and execute code from Internet hosts without explicit approval of the user (e.g., Mosaic, MIME agents).

The TCB for this system provides a single domain for the CAS and another for non-CAS components. At the beginning of each session, the user activates the trusted path to the TCB and then designates the session as a CAS domain session, a non-CAS domain session, or a TCB session. During a CAS session, the TCB allows only CAS programs to run but grants access to both sensitive and non-sensitive files. During a non-CAS session, the TCB allows any program to run, but grants access only to non-sensitive files. In this way, if viruses are present in unscreened programs, the TCB prevents them from damaging files the owner has designated as sensitive. During a TCB session, the user can install programs into the CAS or remove them.

The mandatory and discretionary multiuser access controls, I&A, audit, and other features the TCSEC

requires for TCBs have little relevance to this system. By contrast, the CAS paradigm focuses attention on the system's fundamental security risks and requires only the TCB features that are essential to mitigating them.

5 Discussion

5.1 Covert Channel Strategies

The TCSEC paradigm allows arbitrary programs to access sensitive information and any covert channels that are present within the TCB. TCB developers are supposed to mitigate associated leakage risks by identifying and eliminating covert channels, reducing their capacity, and auditing their use. Unfortunately, the effectiveness of these techniques has been limited, even when the level of effort applied has been substantial. Furthermore, these techniques often impair system performance or curtail system function-

ality. The CAS paradigm suggests an entirely different strategy: deny software of unknown origin or assurance the ability to access sensitive information or leak it. This strategy mitigates covert channel risks by *reducing the likelihood* that an attempt to exploit covert channels will occur or be successful, regardless of the number, capacity, or auditability of the channels present. It cannot be cost effective to require, as the TCSEC paradigm does for high-assurance TCBs, Herculean efforts to identify, reduce, audit, or eliminate covert channels while providing neither motivation nor mechanism for restricting access to the channels that remain.

5.2 Approximating the Idealized Architecture

The objective of the idealized CAS architecture is to place all security-relevant components within a trusted system under the control of the owning organization. Some real-world systems may only be able to approximate the idealized CAS architecture and may not be able to achieve this objective fully. The CAS paradigm is intended to allow for deviations from the ideal and provide insight about the additional risks that may be incurred. Next, we explore the ramifications of relaxing the interface between CAS and non-CAS components.

5.2.1 Adding an API

In the idealized architecture, non-CAS subjects do not exist within sensitive sessions. Moreover, because the CAS does not export a callable³ application program interface (API), non-sensitive sharable objects provide the only interface between these CAS and non-CAS subjects. This interface is highly constrained and is intended to allow only limited importing and exporting of data across the security perimeter. In an MLS system, for example, CAS subjects may be able to read system-low objects created by non-CAS subjects.

The motivation for constraining this interface is to protect sensitive information from being manipulated, even indirectly, by non-CAS programs. To the extent that the interface exported by the CAS becomes more powerful and less constrained, the CAS cedes control over its own sensitive operations to programs that cannot be trusted, even if they're executed by an authorized individual.

In fact, these rules are overly restrictive. Under some circumstances, a CAS can export a highly con-

³We include here a variety of system call mechanisms, including system traps and interprocess communications.

strained callable interface to non-CAS entities that provides no greater power or security risk than the shared-object interface just described. Suppose the CAS in an MLS system were designed to allow a thin layer of non-CAS programs to be interposed between itself and an authorized user during sensitive sessions. If the API exported by the CAS to the non-CAS layer consists of a single callable service that reads non-sensitive files (i.e., reads down), the API conveys no greater risk than the shared file interface. However, if the API also allows reading files at the sensitivity level of the session, the non-CAS layer would be capable of leaking the sensitive information stored in them. In short, providing an API for non-CAS components on top of the CAS is neither inherently insecure nor is it precluded from the paradigm. However, unless such APIs are extremely limited, they can easily introduce vulnerabilities. It is for this reason that they are not depicted in the idealized CAS architecture.

5.2.2 Interpreters in the CAS

Although an OS TCB must prevent CAS subjects from directly executing non-CAS programs, it cannot prevent CAS subjects from indirectly executing them by acting as an interpreter. If a CAS subject acts as an interpreter, it can blur the execution domain boundaries between CAS subjects and untrusted subjects and among CAS subjects in different domains. If the interpreter's command language is sufficiently powerful and it interprets a data file planted by an adversary or incompetent user, it may be subverted. For this reason, there are no interpreters in the idealized CAS architecture.

The fact that the distinction between an interpreter and other kinds of programs is not always clear may make it difficult in some cases to determine whether the CAS is free of interpreters. Many useful programs change their behavior according to tables, macros, or initialization files provided by users and are meant to be tailored by them. On the other hand, there are many conspicuous examples of data-driven systems whose behavior is highly predictable and not subject to security-relevant user tailoring and its accompanying vulnerabilities. The risk that an automated teller machine (ATM) will be reprogrammed from its user interface, for example, is very small. It is entirely feasible to keep many systems, particularly turnkey systems, virtually free of interpreter-related security risks.

The CAS paradigm is intended to address high-assurance CAS domains that need to be free of interpreters and lower assurance CAS domains that may in-

clude interpreters under certain circumstances. Pragmatic measures for mitigating interpreter risks are listed below in order of decreasing potential assurance and increasing flexibility.

- Avoid execution of any program whose behavior cannot be predicted with certainty, particularly programs whose behavior is meant to be tailored by individual users.
- Avoid user-tailorable programs except those that require all tailoring or interpretation instructions to have been installed previously in the CAS; in principle, these have the same assurance as CAS executables.
- Install user-tailorable programs only in domains that can read only high-integrity information [5], that is, information that can be produced only by individuals and programs that can be trusted.
- Install user-tailorable programs only in low-risk domains. Allow only individuals trained to avoid potential vulnerabilities to use these programs.

6 Related Work

This paper is an improved version of an earlier paper presented and used as the subject of a panel session at a recent workshop [31, 32]. Revisions to address issues raised at the workshop include refinements (e.g., regarding the CAS interface and APIs), clarifications, and additional discussion and examples.

The CAS paradigm clarifies, integrates, and extends a number of important ideas in the research literature and restates them in a new context. CAS components and our treatment of sensitive and non-sensitive information generalize the Clark-Wilson integrity model's Transformation Procedure (TP), Constrained Data Item (CDI), and Unconstrained Data Item [7, 8]. For example, a Clark-Wilson TP "must be *certified* to be valid", i.e., a TP must transform "CDIs from one valid state to another." The CAS paradigm, however, allows security functional requirements for CAS components to vary according to the security objective sought and allows assurance procedures to range from formal certification to highly informal processes. Clark and Wilson assert that the confidentiality needs of the military and the integrity needs of the commercial sector are so disparate that they require fundamentally different conceptual models and mechanisms. Instead, we propose a single, unifying paradigm that addresses both. Lee [13] and

Shockley [27] propose implementing TPs as partially trusted subjects whose accesses are constrained according to Clark-Wilson access control tuples by a lattice-enforcing TCB. Although this technique foreshadows the role and use of a TCB within the CAS architecture, neither proponent suggests that the technique is necessary for or applicable to confidentiality; neither acknowledges that application programs in an MLS environment are security relevant.

The CAS architecture builds on previous approaches for layering security mechanisms. Popek and Kline [20] outline an architecture containing multiple "levels of kernels." Shockley and Schell suggest organizing complex TCBs into collections of simpler TCB subsets [28]. Neumann's analysis of hierarchical system architectures for safety, security, and other critical requirements has explored related design and assurance ideas [17, 18]. The CAS architecture allows a CAS or a TCB to be organized internally as a collection of TCB-subset-like layers. Nevertheless, there are important differences between the CAS architecture and the TCB subset approach. In particular, the latter is wed to the TCSEC paradigm and suffers from all of the drawbacks associated with it. Because the TCB subset approach defines security relevance solely in terms of access control, it treats subjects having no special access control privileges as completely innocuous. Consequently, under the TCB subset approach, there is no reason to restrict the interface exported by any TCB subset; in fact, such restrictions would seem objectionable. In contrast, because the CAS paradigm treats subjects that have no special access privileges as potentially harmful, it requires that the interface exported by the CAS to non-CAS subjects be highly restricted; interfaces between layers within the CAS, however, need not.

Fundamental to the CAS paradigm is the balanced assurance philosophy, which arose during the SeaView project [15, 16] and is closely associated with the development of TCB subsets. Other influences on the CAS paradigm include the LOCK system's type enforcement mechanism [6, 35, 19] and other efforts to analyze and automate support for integrity policies [14, 30, 2, 3, 25]; the Military Message System [12], which demonstrated that the trustworthiness of applications can be crucial even for DoD confidentiality policies; Controlled Execution UNIX⁴ [1], a precursor of the CAS architecture that prevents any program that has not been specially installed from being exe-

⁴UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

cuted; and Trusted Mach⁵ [36], whose multiuser server processes clarify the limits of centralized accountability mechanisms.

7 Conclusion

A fundamental assertion underlying the TCSEC paradigm is that all necessary automated security controls for many computer systems can be provided by their operating systems, in particular the components that constitute an OS TCB. This assertion does not hold up in practice because ordinary application processes possessing no special access control privileges can leak sensitive information, undermine an OS TCB's accountability mechanisms, and destroy information integrity. Hence, the security properties ultimately needed by many organizations cannot be enforced by an OS TCB alone and necessarily depend on the benign behavior of application programs.

We have proposed an alternative paradigm based on the notion of a Controlled Application Set (CAS). The CAS paradigm builds on TCSEC principles but identifies and addresses important sources of security risk within trusted systems that are effectively ignored by the TCSEC. For this reason, we believe it can lead to practical improvements in the security of real systems. In addition, as illustrated by the examples above, the CAS paradigm is applicable to a wide range of systems of low and high assurance concerned with a variety of security objectives, including confidentiality, accountability, and integrity.

The CAS paradigm originates from the premise that every software component that can manipulate sensitive information, even if tightly constrained by a TCB, is potentially security relevant. A key implication is that the amount of software on which the security of a trusted system depends will appear in many cases to be much larger than it would under the TCSEC paradigm. The CAS paradigm is an attempt to identify practical techniques for increasing confidence that very large collections of software will behave securely.

The CAS paradigm departs from the TCSEC paradigm in many ways; these have broad implications for trusted systems theory and practice. It charges organizations that own and operate trusted systems with responsibility for controlling the applications used to manipulate sensitive information and, more importantly, provides them with automated en-

forcement mechanisms to prevent other applications from being used for that purpose.

The CAS conceptual architecture and security perimeter provide a new theoretical context for the construction and evaluation of trusted systems. In this context, an OS TCB must be designed and evaluated not as the totality of security protection mechanisms but as the *base* for it. An OS TCB must satisfy or support new requirements, including ensuring that CAS components are tamperproof, non-bypassable, and accessible to users via the trusted path; exporting multiple execution domains so that the CAS can be organized in accordance with the principle of least privilege; and providing the CAS with functionally correct storage and retrieval services. On the other hand, the paradigm diminishes the importance of covert channel elimination, reduction, and auditing requirements and compensates by reducing the likelihood that a malicious agent will be given an opportunity to exploit whatever covert channels are present.

The CAS paradigm relies on balancing assurance requirements pragmatically against risks. Since an OS TCB will address many security risks, the level of assurance needed for CAS components need only be commensurate with the residual risks that remain; in many cases, CAS components may merit significantly less assurance than TCB components. For some CAS components (e.g., COTS products), assurance of benign behavior will be based largely on accountability of origin instead of quality-control spot checks of its behavior or its development history.

Our current research involves building TCBs and prototype extensions that provide much of the support needed for a CAS [2, 3, 25]. We intend to pursue validating the ideas described in this paper through continued prototyping and discussions with practitioners and policy makers in the computer security community.

References

- [1] L. Badger, H. Tajalli, D. Dalva, and D. Sterne. Controlled Execution UNIX. In *Proc. 17th National Computer Security Conference*, pages 254–263, Baltimore, MD, October 1994.
- [2] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghghat. "Practical Domain and Type Enforcement for UNIX." In *Proc. 1995 IEEE Symposium on Security and Privacy*, Oakland, CA, June 1995.

⁵Trusted Mach is a registered trademark of Trusted Information Systems, Inc.

- [3] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat. "A Domain and Type Enforcement UNIX Prototype." In *Proceedings of the 5th USENIX UNIX Security Symposium*, Salt Lake City, UT, June 1995.
- [4] D. Bell and L. LaPadula. *Secure Computer System Unified Exposition and Multics Interpretation*. Technical Report MTR-2997, MITRE Corp., Bedford, MA, July 1975.
- [5] K. Biba. *Integrity Considerations for Secure Computer Systems*. Technical Report TR-3153, MITRE Corp., Bedford, MA, April 1977.
- [6] W. E. Boebert and R. Y. Kain. A Practical Alternative to Hierarchical Integrity Policies. In *Proc. 8th National Computer Security Conference*, pages 18-27, Gaithersburg, MD, September 1985.
- [7] D. Clark and D. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proc. 1987 IEEE Symposium on Security and Privacy*, pages 184-194, Oakland, CA, April 1987.
- [8] D. Clark and D. Wilson. Evolution of a Model for Computer Integrity. In *Proc. 11th National Computer Security Conference*, Baltimore, MD, October 1988.
- [9] H. Custer. *Inside Windows NT*. Microsoft Press, Redmond, Washington, 1993.
- [10] Department of Defense. *Department of Defense Trusted Computer System Evaluation Criteria*, December 1985. DoD 5200.28-STD.
- [11] D. Golub, et al. UNIX as an Application Program. *Proceedings of the Summer 1990 USENIX Conference*, pp. 87-96, June 1990.
- [12] C.E. Landwehr, C.L. Heitmeyer, and J.A. McLean. A Security Model for Military Message Systems. *ACM Trans. on Computer Systems*, Vol. 2, No. 3, August 1984, pages 198-222.
- [13] T.M.P. Lee. Using Mandatory Integrity to Enforce Commercial Security. In *Proc. 1988 IEEE Symposium on Security and Privacy*, pages 140-146, Oakland, CA, April 1988.
- [14] S.B. Lipner. Non-discretionary Controls For Commercial Applications. In *Proc. 1982 IEEE Symposium on Security and Privacy*, pages 2-10, Oakland, CA, April 1982.
- [15] T.F. Lunt, et al. Element-Level Classification with A1 Assurance. *Computers and Security*, 7(1), February 1988.
- [16] T.F. Lunt, et al. A Near-Term Design for the SeaView Multilevel Database System. In *Proc. 1988 IEEE Symposium on Security and Privacy*, pages 234-244, Oakland, CA, April 1988.
- [17] P.G. Neumann. On Hierarchical Design of Computer Systems for Critical Applications. *IEEE Transactions on Software Engineering*, (9):905-920, September 1986.
- [18] P.G. Neumann. *On the Design of Dependable Computer Systems for Critical Applications*. Technical Report SRI-CSL-90-10, SRI International, Menlo Park, CA, October 1990.
- [19] R. O'Brien and C. Rogers. Developing Applications on LOCK. In *Proc. 14th National Computer Security Conference*, pages 147-156, Washington, DC, October 1991.
- [20] G. Popek and C. Kline. The Design of a Verified Protection System. In *Proc. 1974 International Workshop on Protection in Operating Systems*, pages 183-196, Rocquencourt, France, August 1974.
- [21] G. Pottinger. *Proof Requirements in the Orange Book: Origins, Implementation, and Implications*. Mathematical Sciences Institute, Cornell University, Ithaca, NY, February 1994.
- [22] N.E. Proctor and P.G. Neumann. Architectural Implications of Covert Channels. In *Proc. 15th National Computer Security Conference*, pages 28-43, Baltimore, MD, October 1992.
- [23] A.D. Rubin. Trusted Software Distribution Over the Internet. In *Proc. Symposium on Network and Distributed System Security*, pages 47-53, San Diego, CA, February 1995.
- [24] J. Saltzer and M. Schroeder. The Protection of Information in Computer Systems. *Proc. IEEE*, 63(9), March 1975.
- [25] D.L. Sherman, D.F. Sterne, L. Badger, S.L. Murphy, K.M. Walker, S.A. Haghighat. Controlling Network Communication With Domain and Type Enforcement. In *Proc. the 18th National Information Systems Security Conference*, Baltimore, MD, October 1995.

- [26] L.J. Shirley and R.R. Schell. Mechanism Sufficiency Validation By Assignment. In *Proc. 1981 Symposium on Security and Privacy*, pages 26-32, Oakland, CA, April 1981.
- [27] W.R. Shockley. Implementing the Clark/Wilson Integrity Policy Using Current Technology. In *Proc. the 11th National Computer Security Conference*, Baltimore, MD, October 1988.
- [28] W.R. Shockley and R.R. Schell. TCB Subsets For Incremental Evaluation. In *Proc. Third Aerospace Computer Security Conference*, pages 131-139, Orlando, FL, December 1987.
- [29] D.F. Sterne. On The Buzzword "Security Policy." In *Proc. 1991 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1991.
- [30] D.F. Sterne. A TCB Subset for Integrity and Role-Based Policies. In *Proc. 15th National Computer Security Conference*, pages 690-696, Baltimore, MD, October 1992.
- [31] D.F. Sterne, G.S. Benson, and H. Tajalli. Redrawing the Security Perimeter of A Trusted System. In *Proc. Computer Security Foundations Workshop VII*, pages 162-174, Franconia, NH, June 1994.
- [32] D.F. Sterne, G.S. Benson, C. Landwehr, L. Padula, and R. Sandhu. Panel Session: Reconsidering the Role of the Reference Monitor. In *Proc. Computer Security Foundations Workshop VII*, pages 175-176, Franconia, NH, June 1994.
- [33] *Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*. Technical Report CSC-STD-003-85, DoD, June 1985.
- [34] Ken Thompson. Reflections on Trusting Trust. *CACM*, 27(8), August 1984.
- [35] D.J. Thomsen. Role-Based Application Design and Enforcement. In *Proc. of the Fourth IFIP Workshop on Database Security*, Halifax, England, September 1990.
- [36] *Trusted Mach System Architecture*, Technical Report TIS TMACH Edoc-0001-93B, Trusted Information Systems, Inc, Glenwood, MD, May 1993.

INFORMATION DOMAINS METAPOLICY

Gene Hilborn

Computer Sciences Corporation

7471 Candlewood Road

Hanover, MD 21076

Abstract

The metapolicy inherent in the concept of information domains, as used in the emerging Department of Defense Information Systems Security Policy (DISSP) [1], and underlying the Defense Goal Security Architecture [8] is modeled and analyzed. The access control and information transfer metapolicy of the DISSP is formalized as a set of rules that apply axiomatically to all information domain security policies. The relationship between mandatory access control (MAC) and discretionary access control (DAC) system security policies and information domain security policies is analyzed. An information system that enforces a MAC policy is shown to be a highly-structured, special case of the general multiple information domain policy system. Inferences are drawn for the use and limitations of existing MAC/DAC-based systems for implementation of multiple information domain policies. The type of future system features needed to support the full potential of information domain-based multiple security policies is discussed.

1 Introduction

1.1 Background

Multilevel secure systems were developed as a solution to the conflict between computer resource sharing of multiple users and protection of classified information at multiple levels from unauthorized access. The structure of information labeling and user clearances was formulated as an hierarchy or more generally, a partial ordering, or a lattice. An automated information system enforcing a mandatory access control (MAC) policy based on such labeling of information objects and users or subjects acting on their behalf has become the dominant paradigm for "serious" information security, and thoroughly embedded in the technical guidance of the *Trusted Computer System Evaluation Criteria* (TCSEC) [2]. A single system under this paradigm is considered to enforce a single coherent system policy. The single policy may have subpolicy components such as MAC and DAC (discretionary access control), that make up a single, coherent policy. *The Trusted Network Interpretation* (TNI) [3] and *Trusted Database Interpretation* (TDI) [4] further

extend the paradigm to various modes of system and policy composition, but do not depart from the single, global policy and system paradigm.

The still evolving *Common Criteria* [5] contains no rigid policy construct. However, no protection profiles with other than the dominant paradigm have been developed.

Deficiencies in the dominant paradigm have been identified by multiple workers. Hosmer [6] summarized these deficiencies of the single policy paradigm as: its inflexibility to change; the difficulties with data interchange between systems under policy authorities or domains; its unrealistic model of the real world's multiple, sometimes conflicting policy domains; and, its poor performance when manual security guards are introduced to deal with interdomain transfers. As an approach to solving these deficiencies, Hosmer advocated building a "Multipolicy Machine" that enforces multiple, sometimes conflicting security policies through automated metapolicy-enforcing conflict-resolution mechanisms [6]. The problem with this approach is that it is so general and unstructured it is doubtful that the many standardization issues can be resolved in order to reduce it to practice. Bell [7] has developed a framework that abstractly describes such multiple policies, conflicts, and resolutions.

1.2 Information Domains

A new approach to information system policy formulation and subsequent automation was recently initiated in the U.S. Department of Defense (DoD) [1] based on the construct of "information domains." The information domain approach is a significant departure from traditional DoD information system security policies expressed by DAC and lattice-based MAC policies. These (DAC and MAC) policies also form the access control basis for existing evaluated trusted products and systems in accordance with the TCSEC, and its interpretations under the TNI and TDI. While recent, and not widely known or understood, the information domain policy formulation is also a key underpinning of the *Department of Defense (DoD) Goal Security Architecture* (DGSA) [8]. As important as the information domain policy approach is as a foundation of the DGSA, it has not been rigorously

formulated or modeled in published work, and has not yet formed the basis of any available trusted products.

While the use of the information domain approach originated as US DoD policy, it is potentially more applicable to commercial environments than are the traditional lattice-based MAC policies, which have not been widely accepted in the commercial market.

1.3 Goals and Limitations of the Paper

The goals of this paper are to:

- Stimulate wider exploration and analysis of the information domain policy idea,
- Provide a mathematically formalized basis for statement of information domain security policies,
- Examine the relationship of information domain policies to traditional MAC and DAC system policies.
- Explore implications for existing and future trusted products and systems.

The formalization of information domain metapolicy is done using sets and functions to express a set of rules about objects, accesses, and interdomain information transfer. This process provides a basis for consistent policy formation, and illuminates the power and limitations of the information domain construct.

The scope of the paper is limited to information access aspects of the DISSP [1]. Other aspects of the DISSP, such as protection and strength of mechanisms are not formalized or analyzed. The DGSA [8] is discussed only as it interprets the DISSP information domain metapolicy.

2 Informal Definitions

According to [1], an *information domain* combines the following:

- A set of *information objects*, identifiable as belonging to the domain
- A set of (human) *members* of the domain
- An information domain *security policy* that includes:
 - the requirements for membership
 - the rules of access by members to information objects of the domain
 - the rules of import and export of information from/to other information domains

- the required protection of the information objects of the domain

To promote consistency, interoperability, and trusted products that support multiple information domains, constraints are imposed on the nature of information domain security policies. The *Department of Defense Information Systems Security Policy* [1] states an overall DoD policy explicit on the minimum constraints imposed by the information domain idea itself and additional policy that the DoD imposes on each of the information domains under its jurisdiction.

The following informally summarizes the author's interpretation of additional information domain policies which are considered to be inherent in the information domain idea, independent of other policy.

- a. All information objects in an information domain have identical security attributes.
- b. All members of an information domain need not have equal access to its information objects.
- c. A given member has identical access rights to all information objects in an information domain.
- d. No information object belongs to more than one information domain.
- e. Individuals may be members of more than one information domain.
- f. Transfer of information between domains occurs only in accordance with the policies of both the exporting and importing domain.
- g. Transfer of information between information domains can be accomplished only by a member of both the exporting and importing domains.
- h. Protections requirements for an information domain are stated independently of any other information domains.

It is implicitly assumed that:

- i. Only the members of an information domain have access to its information objects.
- (a) and (c) are interpreted in the DGSA [8] to be equivalent.

3 Information Domain Metapolicy

3.1 Information Domain Definition

An information domain D is defined as a triple of information objects, members, and a policy. Symbolically,

$$D = (O, M, P).$$

Strictly speaking, this formulation is static, which means that any change in the sets of members or objects would change the information domain. Real information domains need to provide for the admission and exit of members, and the creation and deletion of information objects, with persistence of the named information domain. A more elaborate formulation would incorporate a dynamic structure for members and objects of an information domain, e.g., by defining an equivalence class. This potential refinement is omitted in the present formulation.

3.2 Single Information Domain Metapolicy

Let A represent the set of access modes possible for the information objects in a domain, (e.g., read, delete, append, modify, etc.). Many security models describe a current security state by an access function that maps object-subject pairs to subsets of A . This kind of access function is an *access state* function. At any one time an access state function represents the existing or granted accesses of subjects to objects. The potential or allowable accesses of subjects to objects can also be modeled as a function mapping object-subject pairs to subsets of A . Such a mapping is an *access rights* function. The difference between an access state function and an access rights function is that the later is a static expression of policy, and represents all allowable accesses, whether or not they are in current use. The access rights function, α for information domain D assigns a subset of A to each (information object, member pair). Symbolically,

$$\alpha: O \times M \rightarrow 2^A,$$

where 2^A denotes the set of all subsets of A .

The constraint on policy that all information objects in a domain have identical security attributes can be expressed concisely in terms of member access as follows:

Rule 1 (Object-Independent Access): For an information domain $D = (O, M, P)$, the policy P restricts the access rights function α such that for any $m \in M$, and any two objects $o_1 \in O$ and $o_2 \in O$,

$$\alpha(o_1, m) = \alpha(o_2, m).$$

Thus the structure of access rights permitted by the security policy of an information domain is very simple. If the access rights function is expressed as a matrix with members identified with rows and information objects identified with columns, then all columns must be equal. The access rights within an information domain can also be described by subsets of the members who have the same access rights, without reference to information objects. Since the access rights function α is independent of objects, it can be

replaced by a *member access rights function*, ζ with M as its domain of definition:

$$\zeta(m) = \alpha(o, m),$$

where o is an arbitrary object in O .

3.3 Multiple Information Domain Metapolicy

Let D_1, D_2, \dots, D_N , where $D_i = (O_i, M_i, P_i)$, be a finite set of information domains. O and M are the total sets of information objects and members, respectively, and O_i and M_i are subsets of O and M , respectively.

That each information object belongs to a single information domain is expressed as follows:

Rule 2 (Information Object-Isolation): For distinct information domains, D_1, D_2, \dots, D_N , where $D_i = (O_i, M_i, P_i)$, for all $1 \leq i \leq N$ and $1 \leq j \leq N$, if $i \neq j$ then

$$O_i \cap O_j = \emptyset.$$

Since information objects are containers of information, Rule 2 says nothing about the information *content* of the objects, which could well be duplicated across information domains.

Since Rule 2 categorizes every information object as belonging to a single, distinct information domain, there can be no such thing as a "multidomain information object" (meaning an information object that is marked as belonging to multiple information domains). However, this constraint does not prohibit the simultaneous access by a member of multiple information domains to objects in those different information domains. Such a simultaneous access could be used, for example, to construct a display that has the "look and feel" of a "virtual multidomain information object."

The following constraint formalizes the idea that only the members of an information domain may have access to its objects.

Rule 3 (Member-Only Access): For distinct information domains, D_1, D_2, \dots, D_N , where $D_i = (O_i, M_i, P_i)$ with member access rights function ζ_i , $1 \leq i \leq N$, and any $m \in M$,

$$\zeta_i(m) \neq \emptyset \Rightarrow m \in M_i.$$

In describing access rights in a multiple information domain context, the access mode set A is the collection of all the access types needed in the various information domains under discussion, even though some types may not be used in a particular information domain. For an information domain whose policy defines conditions for the export of information to

another information domain, A contains an export mode of access, symbolized E . Similarly, an information domain that permits the import of information from another information domain, A contains an import mode I . The description of export in terms of information rather than the export of information objects is consistent with the simplified static model of information domains, each having a fixed number of information objects. The right to transfer information is modeled by an E access (which includes read) for an information object in the originating domain and an I access (which includes modify or append) to an information object in the destination domain.

The security policy of an information domain establishes conditions for import and export, such as which members have the right to export to which other information domains. The domain's security policy could also establish other import/export conditions provided they do not violate Rule 1.

Let the members M_i of information domain D_i who are permitted by the policy P_i to export from D_i to D_j be denoted by $M_i(E_j)$. Similarly, let the members of M_i who are permitted by P_i to import to D_i from D_j be denoted by $M_i(I_j)$.

When the transfer of information directly from information domain D_1 to a different information domain D_2 is allowed by their combined policies, D_1 is said to be *adjacent* to D_2 , symbolized " $D_1 > D_2$ ". Adjacency is directed; $D_1 > D_2$ does not imply $D_2 > D_1$. The use of the term "directly" means that no other information domain is required for the transfer. Symbolically,

$$D_1 > D_2 \Leftrightarrow M_1(E_2) \cap M_2(I_1) \neq \emptyset.$$

It follows from Rule 3 that $M_1(E_2)$ is a subset of M_1 , and that $M_2(I_1)$ is a subset of M_2 . Therefore $M_1(E_2) \cap M_2(I_1)$ is a subset of $M_1 \cap M_2$. The necessary and sufficient conditions for direct information transfers can therefore be stated as follows:

Rule 4 (Inter-information domain transfers):

Information domain D_1 is adjacent to information domain D_2 , if and only if there is at least one member of both information domains, who is permitted by the policy of D_1 to export information to D_2 , and is permitted by the policy of D_2 to import information from D_1 .

An information domain with no adjacency to any other information domain is *isolated*.

It is possible for the policy of an information domain to vest members with import or export authority, but for the information domain to be isolated because no member also has compatible import or export authority in another domain.

Even when information domains are non-adjacent, transfer of information can be accomplished indirectly by using one or a chain of intermediary information domains that form a directed graph chain of adjacent information domains. When no such chain exists in either direction, two information domains are *pairwise isolated*.

Suppose for example, the members of information domains D_1 and D_2 with no members in common decide they want to make controlled transfers of information from D_1 to D_2 , and they want to continue to have no members in common, they can create a third shared information domain D_3 , such that $D_1 > D_3$ and $D_3 > D_2$. To satisfy Rule 4, there is at least one member of D_1 who can export to D_3 and one member of D_2 who can import from D_3 , i.e.,

$$M_1(E_3) \cap M_3(I_1) \neq \emptyset$$

and

$$M_3(E_2) \cap M_2(I_3) \neq \emptyset.$$

An example adjacency graph of four information domains is illustrated in Figure 1. In this example, D_1 and D_2 are not adjacent, but information may be transferred indirectly via D_3 . D_4 is pairwise isolated from each of the others, and therefore isolated.

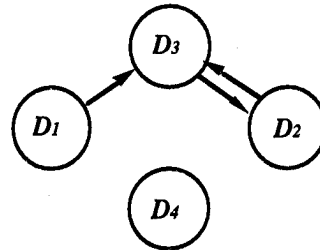


Figure 1. Information domain adjacency graph.

3.4 Information System Security Policies

Information domain security policies have been abstractly formulated in terms of information objects and member access rather than in terms of system behavior. Information systems can support one or more information domains. The question then arises of what is the distinction and relationship between information system security policies and information domain security policies.

For an automated information system supporting a single information domain, an information domain security policy that is automated by the system is the same as the system security policy. For a system supporting multiple information domains, the security policies of all the supported information domains must be supported (enforced). Such a system is more properly called "multipolicy" secure than "multilevel" secure, since a multilevel secure system enforces a single security policy with certain rules of access based on object and subject "levels." "Multipolicy secure" (MPS) is also more appropriate than "multilevel secure" (MLS) because there is no requirement for information domains to have any particular relationship as "levels" with a partial ordering or lattice. As will be illustrated in the next section, a system that is MLS is a special case of system that is MPS. In summary, the system security policy of a multiple information domain system is the combined enforcement of all the policies of the individual information domains supported.

4 MAC and Information Domains

4.1 MAC Security Policies

Mandatory Access Control (MAC) policies are characterized as follows. There is a set L of N distinct sensitivity levels:

$$L = \{L_1, L_2, \dots, L_N\}.$$

There is a set of information objects O , and a set of subjects M . The abstraction "subject" typically represents a user or processing on a user's behalf at a given level. There is an assignment ω of sensitivity level to each information object $o \in O$:

$$\omega: O \rightarrow L.$$

There is an assignment λ of sensitivity level to each subject $m \in M$:

$$\lambda: M \rightarrow L.$$

There is an access mode set A that contains modes R —representing read-equivalent access (e.g., view, copy-from), and W —representing write-equivalent access (e.g., modify, append, clear). The first principle of MAC is that the access rights function

$\alpha: O \times M \rightarrow 2^A$ can always be expressed through a function of the object and subject sensitivity levels. There is a function f such that

$$\alpha(o, m) = f(\omega(o), \lambda(m)),$$

where

$$f: L \times L \rightarrow 2^A.$$

Since α is not independent of information objects, it does not satisfy Rule 1 for information domains. Therefore (O, M, P) , where P specifies such an access

rights function, can not be an information domain. However it is possible to find a set of N embedded information domains D_1, D_2, \dots, D_N that together comprise the same access policy.

The second principle of MAC is that if any information may flow between an object and a subject at different levels, it may only flow "upward." "Upward" is expressed in terms of a partial ordering \geq on L . The partial ordering operator \geq satisfies the three axioms of idempotency, reflexivity, and transitivity. When $x \geq y$, x is said to dominate y . When $x \geq y$ and $x \neq y$, x is said to strictly dominate y , as indicated by $x > y$.

The MAC policy that is least restrictive on information flow between levels is "read-down/write-up," (also called simple-security/*-property in the Bell and LaPadula model [10]. For read-down/write-up,

$$\alpha(o, m) = \begin{cases} \{W\}, \omega(o) > \lambda(m) \\ \{R, W\}, \omega(o) = \lambda(m) \\ \{R\}, \lambda(m) > \omega(o) \\ \emptyset, \text{ otherwise.} \end{cases}$$

A strictly dominated write-up is sometimes unacceptable from either a policy viewpoint or an implementation viewpoint. An example of a policy issue is the integrity requirement to protect high-level information from corruption by low-level subjects, who are not allowed to see any modifications they are making. An example of an implementation issue is the infeasibility of performing write without read operations on some types of information objects. To address these difficulties, many MLS systems implement a read-down/write-equal variant of MAC policy. This variant of MAC simply restricts the access rights function by eliminating the strictly dominated write-up access. Thus for read-down/write-equal,

$$\alpha(o, m) = \begin{cases} \{R, W\}, \omega(o) = \lambda(m) \\ \{R\}, \lambda(m) > \omega(o) \\ \emptyset, \text{ otherwise.} \end{cases}$$

An example of an MLS system enforcing a read-down/write-equal policy is the Compartmented Mode Workstation [11].

When the access rights function is restricted to also eliminate read-down, the MAC policy reduces to read-equal/write-equal or level-isolation. For level isolation,

$$\alpha(o, m) = \begin{cases} \{R, W\}, \omega(o) = \lambda(m) \\ \emptyset, \text{ otherwise.} \end{cases}$$

For the level-isolation variant, no partial ordering among the sensitivity levels is required. An example

of an MLS system implementing a level-isolation policy is the Multinet Gateway [12].

4.2 Multiple Information Domain Policies Corresponding to a Single MAC Policy

To demonstrate how single MAC policy is re-stated in terms of a set of information domains and their policies, define N sets of information objects as follows:

$$O_i = \{o \in O: \omega(o) = L_i\},$$

for $1 \leq i \leq N$.

Since the levels are distinct, these sets of information objects are disjoint and therefore satisfy Rule 2 for information domains. The members of these information domains are to be identified with MAC subjects.

Next, segment the MAC access rights function α into N information domain member access functions ζ_i :

$$\begin{aligned} \alpha(o, m) = & \zeta_1(m), o \in O_1 \\ & \zeta_2(m), o \in O_2 \\ & \dots \\ & \zeta_N(m), o \in O_N \end{aligned}$$

Since ζ_i is dependent only on subjects/members, there is an access rights function α_i equal to ζ_i for each i that satisfies Rule 1 for information domains, and expresses the access policy of each information domain.

Define the following subsets of M :

$$\begin{aligned} M_i(+) &= \{m \in M: \lambda(m) > L_i\} \\ M_i(0) &= \{m \in M: \lambda(m) = L_i\} \\ M_i(-) &= \{m \in M: L_i > \lambda(m)\}. \end{aligned}$$

For each $1 \leq i \leq N$, these three sets are disjoint.

For a read-down/write-up MAC policy, the members of information domain D_i are

$$M_i = M_i(+) \cup M_i(0) \cup M_i(-), \text{ for } 1 \leq i \leq N.$$

Let

$$\begin{aligned} \zeta_i(m) = & \{R\}, m \in M_i(+) \\ & \{R, W\}, m \in M_i(0) \\ & \{W\}, m \in M_i(-). \end{aligned}$$

It follows that for each $1 \leq i \leq N$, $D_i = (O_i, M_i, P_i)$ is an information domain where policy P_i permits

member access rights ζ_i defined above. Each of these information domains has three kinds of members within M_i . The $M_i(+)$ members are those who have an associated level (e.g., "clearance" or login level) that strictly dominates L_i , and who have read-only access to the information objects O_i . The $M_i(0)$ members are those who have an associated level equal to L_i , and who have read and write access to the information objects O_i . The $M_i(-)$ members are those who have an associated level that is strictly dominated by L_i , and who have write-only access to the information objects O_i .

For a read-down/write-equal MAC policy, the members of information domain D_i are

$$M_i = M_i(+) \cup M_i(0).$$

Let

$$\begin{aligned} \zeta_i(m) = & \{R\}, m \in M_i(+) \\ & \{R, W\}, m \in M_i(0). \end{aligned}$$

The corresponding information domains $D_i = (O_i, M_i, P_i)$ for $1 \leq i \leq N$ are information domains with two kinds of members. The $M_i(+)$ members are those who have an associated level ("clearance") that strictly dominates L_i , and who have read-only access to the information objects O_i . The $M_i(0)$ members are those who have an associated clearance level that is the same as L_i , and who have read and write access to the information objects O_i .

For a level-isolation MAC policy, the members of information domain D_i are

$$M_i = M_i(0).$$

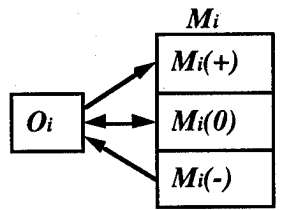
Let

$$\zeta_i(m) = \{R, W\}, m \in M_i(0).$$

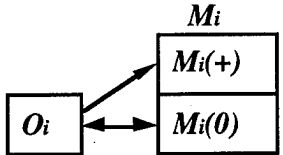
$D_i = (O_i, M_i, P_i)$ is an information domain with all the members having an associated clearance level of L_i , and all having read and write access rights to the information objects L_i .

The information-access relation of the information domain members to the information objects is illustrated in Figure 2 for each of the information domains imbedded in each of above three variants of MAC. An arrow from object to subject indicates read access is permitted; an arrow from subject to object indicates write access is permitted; and an arrow

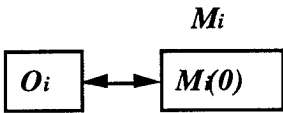
pointing both ways indicates read and write are both permitted.



(a) Read-Down/Write-Up



(a) Read-Down/Write-Equal



(c) Level Isolation

Figure 2. Access of members to information objects in information domains embedded in MAC policies.

4.3 Adjacency of MAC-Based Information Domains

Since only read and write equivalent accesses were defined to characterize MAC policies, "export" access is considered to be read-equivalent, and "import" access is considered to be write-equivalent. The members of D_i who are permitted to export to D_j are those who are permitted to read in D_i . Similarly, the members of D_i who are permitted to import from D_j are those who are permitted to write in D_i . Symbolically,

$$M_i(E_j) = \{m \in M_i \mid R \in \zeta_i(m)\},$$

and

$$M_i(I_j) = \{m \in M_i \mid W \in \zeta_i(m)\}.$$

For the read-equal/write-equal MAC policy variant, all the composing information domains are isolated. For read-down/write-equal, read-equal/write-up, and read-down/write-up the information domains are adjacent whenever the corresponding levels have a (strict) dominance relationship, i.e.,

$$D_i > D_j \Leftrightarrow L_j > L_i.$$

What differs between the information domains embedded in variants of MAC that have adjacencies is who can perform the transfer of information between the information domains. When the MAC policy permits read-down/write-up, then any member m whose clearance level is bracketed by the levels of the exporting and importing information domains will be a member of both and allowed to perform transfers:

$$\lambda(m) \geq L_i$$

and

$$L_j \geq \lambda(m).$$

On the other hand, for a read-down/write-equal MAC policy, the member clearance level must equal that of the importing information domain and strictly dominate that of the exporting information domain:

$$\lambda(m) = L_i$$

and

$$L_j > \lambda(m).$$

A system that implements a MAC policy is thus capable of supporting multiple embedded information domains, provided the information domains are either isolated, or related through a partially ordered set of sensitivity labels. For other than isolation MAC policy, the adjacency graph of the set of embedded information domains is isomorphic to the partial ordering graph of the sensitivity levels.

An adjacency graph for four information domains in a lattice relationship is illustrated in Figure 3. In the illustrated set of information domains, L_4 is the Zero element (dominated by all) of the lattice, L_3 is the Unit element (dominates all) of the lattice, and L_1 and L_2 are in between with no dominance relationship between them.

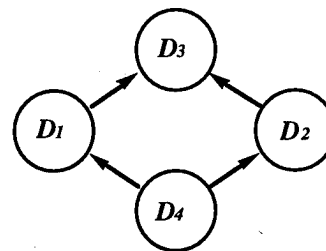


Figure 3. Lattice-related information domain adjacency graph.

4.4 Uses and Limitations of MAC Based Systems

As demonstrated above, MAC is a special case of the kinds of relationships that can exist between information domains. A system that enforces a label-based MAC policy (e.g., TCSEC B1 and higher systems) is capable of supporting multiple information domains when these information domains are either isolated or can be related by a partial ordering or lattice.

While most MAC trusted products nominally enforce either a read-down/write-up or a read-down/write-equal MAC policy with members of the level set each composed of a hierarchical level and a set of non-hierarchical categories. However such systems can also be effectively set up to enforce isolation of N information domains as follows. Let each information domain correspond to a non-hierarchical category, and define or use only a single hierarchical level whose name is unimportant. In addition to the N information domains corresponding to the N non-hierarchical categories, define a system-low (no categories) public information domain, and system-high (all categories) information domain. The public information domain provides such public information as executable software for general use. The system-high information domain is for system security administrative use, e.g., auditing.

There can be highly useful multiple information domains that are not expressible as imbedded in any MAC policy, and therefore not supported by an existing MAC system. A very simple and yet clearly useful example is where two information domains D_1 and D_2 have no members in common, but need to make controlled transfers of information to one another (e.g., two businesses or two government agencies with different missions and people). Members the two information domains agree to create two new information domains each D_3 and D_4 that each have members from both D_1 and D_2 . Some members of D_1 may export to D_3 and members of D_2 may import from D_3 . Members of D_2 may export to D_4 and members of D_1 may import from D_4 . The information objects of D_3 could be a mail queue that holds information released from D_1 and destined only to D_2 . Similarly, the information objects of D_4 could be a mail queue that holds information released from D_2 and destined only to D_1 . The adjacency graph (Figure 4) is cyclic, and could not therefore correspond to any MAC policy, since a cyclic graph is not isometric to any partial ordering.

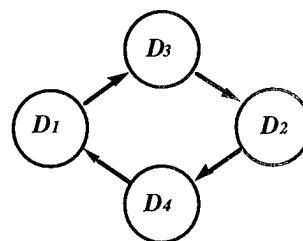


Figure 4. Cyclic information domain adjacency graph.

5 DAC and Information Domains

5.1 DAC Security Policies

Discretionary access control (DAC) policies [2] permit assignment of access rights of system users to information objects on a system at the discretion of the "owner" of each information object. In general the accesses rights permitted by the owner of an information object may be changed at any time. The corresponding access rights matrix can be interpreted in information domain terms as either highly dynamic and object dependent, or as "weak" (in that every entry equals the total access modes set A because all accesses are permitted under some owner decisions).

5.2 DAC and Information Domains

Since DAC is a much less rigid concept than MAC, there are several different mappings that can be made between DAC and information domain policies, depending on the DAC-interpretation adopted.

Under a "weak" interpretation, a system that implements a DAC policy, can support only a single information domain, where all the users are its members, who have (potentially) equal access rights to all information objects. In this interpretation, the individual owner-assignments and revocations of access permission are not relevant from an information domain policy viewpoint. They are a functional convenience to the members of the information domain to manage their activities.

Alternately, a dynamic interpretation could be made where there are as many information domains on a system as information objects. The members of a single information domain are all those who have any access assigned by the owner. Since the owner can change these permissions at-will, the membership changes with each such change. Each such change changes a column of the access rights matrix.

On the other hand, a system DAC policy can have associated procedural rules or other mechanism so that it is neither weak nor dynamic. For example, if the only owner of information objects is a security administrator, the administrator can use the DAC mechanism to group information objects and users into multiple information domains. If an access control list

(ACL) mechanism is used, then the objects in a single information domain are all those with the same ACL. (An ACL is equivalent to a column of the access rights matrix.) To satisfy Rule 1, the information objects of a single information domain are those with identical ACLs. All the users who are assigned one or more access modes in that ACL are its members. While such a "strong" DAC policy could enforce any information domain access control policy, it may not be acceptable for other reasons such as the protection weakness inherent in the all-powerful nature of the administrator across all information domains.

In a system that provides MAC and typically "weak" DAC enforcement mechanisms, the MAC mechanism can be used to establish rigid information domain boundaries (within the limitations of MAC), and the DAC mechanism can be used to provide a convenience for system users to manage their information within each information domain, independently of the formal information domain security policy.

6 Interconnected MPS Systems

All information systems supporting the same information domain must be compliant to its security policy, including access controls and protection mechanisms. These systems may or may not be directly or indirectly connected. Establishing interconnections or security association between these systems can provide a mechanism for information transfer only within the same information domain and therefore in accordance with the same policy. The idea of "connective association" includes both continuous or interactive connection or discrete, staged, or connectionless information transfer.

If the ability to enforce interdomain transfer policy in accordance with Rule 4 is enforced by some form of reference monitor, then presumably such transfers can occur only within a single information system. Under this assumption (which is a requirement of the DGSA [8]), it follows that transfer of information between information domains in accordance with both domain policies can only occur on a system that enforces both policies. Thus by connective associations, information domains can extend across multiple systems in any combination of systems and information domains, provided (a) each information domain's security policy is enforced by the supporting systems, and (b) transfer of information between domains occurs only on systems that support both the exporting and importing domains.

A significant difficulty that accompanies the traditional one-system/one-policy paradigm is the celebrated "composition problem" [13, 14]. The traditional composition problem formulation merges system boundaries. Interconnection of two systems where each enforces a policy of its own is viewed as creating a composite system with functionality allowed by the

interconnection, and security properties that enforce a composite single policy [13]. The information domain formulation essentially "sidesteps" this aspect of the composition problem. If all connected systems support information domains as constrained by information domain metapolicy, their connection raises no new policy "composition" issue. There is no need to redefine or merge system boundaries; each system maintains its identity. Of course there are other significant composition issues, to be solved such as assurance, strength of mechanism, and accreditation.

7 General MPS Systems

While MAC-enforcing MLS systems can support special kinds of multiple information domain policies, with significant levels of assurance, there are currently no trusted products that support more general multipolicy systems with an information domain metapolicy. The DGSA [8] advocates creation of information systems that deal with this problem by extending the reference monitor idea to that of separation of policy enforcement mechanisms from policy decision mechanisms. Such an approach extends the separation kernel idea of Rushby [15] by creating a security context for each information domain where its policy is enforced.

The separation of policy decisions and policy enforcement is roughly as follows: Associate an information domain identifier with each information object. Associate a set of information domain memberships with each user or user-subject. Associate a subset of current information domain identifiers with each active subject (e.g., process) operating on behalf of a user. Each attempted access between subject and object is mediated by an enforcement mechanism that in turn calls on a policy decision mechanism that returns an access decision based on the access policy of the information domain identified with the object and the information domain or domains identified with the subject.

Potentially, each information domain policy could be changed independently of each other and of the enforcement mechanism. Before it becomes practical to "plug-in" a policy for each information domain in a multipolicy machine, a standardized scheme of encoding information domain identities and policies is needed.

8 Summary and Conclusions

The information domain metapolicy described provides a consistent framework for the coexistence of a set of different security policies in multipolicy systems. This framework is intermediate between the rigid structure of a mandatory access control policy based on lattice or partial ordering, and an arbitrary collection of multiple policies with no basis for consistent interaction or shared enforcement support mechanisms and systems.

The metapolicy provides a consistent basis for transfer of information between information domains in mutual accordance with their policies, without any requirement for hierarchical or partial ordering relationships.

The access control aspects of the information domain concept inherent in the emerging DoD information systems security policy [1] were reduced to a set of four axiomatic metapolicy rules. This formalization provides a basis for consistent multiple information domain policy formation as well as insight into the power and limitations of this security policy framework.

The formalism introduced enabled demonstration of how MAC policies can be expressed as a special case of multiple information domain policies. The mapping of DAC policies to information domains metapolicy is less fixed, as there are many types of DAC. The only clear mapping of a system policy supporting only "weak" DAC is to a single information domain. When a system supports both DAC and MAC the MAC mechanism can be used to establish the limited kinds of information domains, and the DAC mechanism can be considered to be unrelated to information domain policies.

While there are currently no general multipolicy trusted products, the information domain approach provides a metapolicy framework in which such products could be built. Moreover it provides the basis for confident system interconnection that sidesteps the access policy composition problem.

References

1. *Department of Defense Information Systems Security Policy*, DISSP-SP.1, 22 February 1993.
2. *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.
3. *National Computer Security Center, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria (TNI)*, NCSC-TG-005, July 1987.
4. *National Computer Security Center, Trusted Database Management Interpretation of the Trusted Computer System Evaluation Criteria (TDI)*, NCSC-TG-021, Version 1, April 1991.
5. *Common Criteria for Information Technology Security Evaluation*, CCEB-94/082, Version 0.9, October 1994.
6. Hosmer, Hillary H., "The Multipolicy Paradigm," *Proceedings of the 15th National Computer Security Conference*, October 1992, Baltimore, MD, pp. 409-422.
7. Bell, D. Elliott, "Modeling the 'Multipolicy Machine'," *Proceedings of the New Security Paradigms Workshop*, August, 1994, pp. 2-9.
8. *Department of Defense (DoD) Goal Security Architecture (DGSA)*, Center for Information System Security Program, Version 1.0, 1 August 1993.
9. *Security Requirements for Automated Information Systems (AISs)*, DoDD 5200.28, March 21, 1988.
10. Bell, D. E. and LaPadula, L. J., *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR-2997 Rev. 1, MITRE Corp., Bedford, Mass., March 1976
11. *National Computer Security Center, Final Evaluation Report SecureWare, Incorporated Compartmented Mode Workstation Plus*, CSC-EPL-91/002, 30 January 1991.
12. Freeman, J., Neely, R., and Dinolt, G., "An Internet System Security Policy and Formal Model," *Proceedings of the 11th National Computer Security Conference*, 1988, pp. 10-19.
13. Tinto, Mario, *The design and Evaluation of INFOSEC Systems: The Computer Security Contribution to the Composition Discussion*, National Computer Security Center C Technical Report 32-92, June 1992.
14. King, Guy., "The Composition Problem: An Analysis," *Proceedings of the 17th National Computer Security Conference*, October 1994, Baltimore, MD, pp. 292-297.
15. Rushby, John, "A Trusted Computing Base for Embedded Systems," *Proceedings of the 7th DOD/NBS Computer Security Symposium*, pp. 294-311.

Maintaining Secrecy and Integrity in Multilevel Databases: A Practical Approach

Sushil Jajodia* Don Marks† Elisa Bertino‡

Abstract. In a multilevel database, certain integrity constraints create a secrecy problem since they cannot be evaluated without access to data at higher classifications than the classification of the data to be modified. We present a practical approach for enforcing such constraints without sacrificing the secrecy requirements. Our approach requires that the such constraints be rewritten as a collection of level-valid constraints. Level-valid constraints meet the secrecy requirement since their evaluation does not require access to any data that is classified higher than the classification of data to be modified. Moreover, they meet the integrity requirements since any database state that satisfies the level-valid constraints satisfies the original constraints as well. The cost associated with this approach is that trusted processes must be relied upon to make occasional modifications.

1 Introduction

Consistency is an important property of a database. One way to achieve consistency is to associate with each database a set of *integrity constraints*. Database management system (DBMS) has the responsibility to ensure that these integrity constraints are satisfied by the database state at all times. A multilevel secure (MLS) DBMS has the additional responsibility of preventing improper disclosure[§] of information either by direct or indirect means. Direct violations are eliminated by enforcing “no read up” and “no write down” requirements on all subjects. Indirect means of illegal information leakages such as those via covert channels (signaling or timing channels) are more difficult to prevent.

It is well-known that there are inherent conflicts in MLS databases between the secrecy requirements and certain types of integrity constraints [Den86]. In particular, it is not possible to enforce certain integrity constraints without violating the secrecy requirements. To illustrate, consider a database consisting of two relations as follows: $EMP(ename, mname, salary)$ that contains for each employee his name, the name of his manager, and his salary and $MGR(mname, salary)$ that contains the name and salary of each manager. Suppose that the EMP relation is considered Low, while the MGR relation is considered High. Say MLS DBMS must enforce the integrity constraint I that requires that an employee cannot have higher salary than that of any manager. We call I a *multilevel-valid* constraint since to verify if the Low data can be modified, both High and Low data need to be accessed.

Every time a new tuple is to be inserted to the EMP relation, constraint I needs to be checked. Unfortunately, this simple integrity constraint presents a dilemma to the MLS DBMS. Since the tuple is being inserted into a Low relation, the transaction T performing the insertion must be considered a Low transaction, in which case T will not have read access to High MGR relation according to the simple security restriction on T . As a consequence, DBMS will not be able to enforce the integrity constraint. Even if we were to assume that MLS DBMS gives T the read access to the salaries in the High MGR relation, MLS DBMS still cannot force T to abort whenever the

*Department of Information & Software Systems Engineering and Center for Secure Information Systems, George Mason University, Fairfax, VA 22030-4444, U.S.A. The work of S. Jajodia was partially supported by National Security Agency under grant MDA904-94-C-6118 and by National Science Foundation under grants IRI-9303416 and INT-9412507.

†Office of the INFOSEC Computer Science, Department of Defense, Ft. Meade, MD 20755, U.S.A.

‡Dipartimento di Scienze dell'Informazione, Università di Milano, 20135 Milano, Italy

§In this paper, we address only mandatory access controls

insertion by T fails to satisfy the given integrity constraint I . This is because doing so would open up a signaling channel which could easily be exploited by Trojan horses.

Since the enforcement of multilevel-valid constraints involves a trade-off between secrecy and integrity, the usual approach is to accept one or another. If secrecy is strictly enforced, multilevel-valid constraints cannot be enforced. On the other hand, if the multilevel-valid constraints are enforced, then signaling channels can be used to subvert the secrecy policy.

In this paper, we show how it is sometimes possible to maintain secrecy while enforcing integrity at the same time. The approach taken will be to translate the original multilevel-valid constraint whose satisfaction requires MLS DBMS to access both High and Low data into a collection of *level-valid* constraints; each level-valid constraint has a fixed security level associated with it and can be evaluated by referencing only data at or below that level. Such constraints will specify conditions where a process may modify the database without compromising either the secrecy or the integrity of the data. Secrecy will not be violated since a level-valid constraint, by design, references data at or below the level of the constraint. Integrity will be preserved since the level-valid constraints are derived from the multilevel-valid constraints in such a way that any allowable database state (i.e., a state satisfying the level-valid constraints) will automatically satisfy the multilevel-valid constraint.

Of course, all this will not come for free. There may exist database states which are allowable under the original multilevel-valid constraints, but not under the derived level-valid constraints. Processes allowed to modify the database in a way that meets the original multilevel-valid constraints, but not the appropriate level-valid constraint, will have to be trusted. Thus, our approach can be viewed as a compromise between the two extremes. That is, both secrecy and integrity will be guaranteed to hold if we occasionally rely on trusted processes to make modification to the database state.

2 Terminology

The "multilevel secure" model classifies data at various levels, such as U (unclassified), C (confidential, the lowest), S (secret) or TS (top secret, the highest), and users are cleared to similar levels. Usually, we will simply use the designations "High" and "Low" to indicate the relative level of clearances or classifications for the two levels of data being compared. The security policy consists of two requirements, "no read up" and "no write down". A C user is allowed to read only U and C data, an S user may read U, C, or S data, a TS user may read any data. In the BLP model, a TS user may only write TS data, an S user may write S or TS data, while a U user may write data at any level.

While humans must be trusted not to read High classified data and then pass that along to lower cleared people, computer processes cannot be so trusted. The restriction on writes by C, S, and TS users therefore addresses the problem of a "Trojan horse", a process that performs unauthorized functions. For example, a High cleared process, possibly unknown to the High users, might read information and then write that information into a Low classified area, passing the information to the lower cleared users. This would violate a basic tenet of secure systems that information cannot be passed to lower users without specific authorization (i.e. "downgrading").

It may be possible for a High process to communicate High information by means other than simply writing into a file visible to the Low users. Such means are called "covert channels" and are just as objectionable as direct writing. Occasionally High cleared processes have legitimate reasons for transmitting information to lower cleared users. Processes allowed to do such writing down are said to be "trusted." The term trusted therefore implies more than simply guaranteeing that the write down is authorized, it guarantees that the intended write down is all that is done (i.e., the code does not contain any trojan horses).

3 Related Work

Although the issue of conflict between the multilevel security and database integrity requirements has been raised by several researchers [Den86, AD87, MJ88, Bur90], no one has developed an approach for enforcing multilevel-valid constraints in a secure manner.

Recently, Thuraisingham [Thu91], Smith and Winslett [SW92], and Qian [Qia94a, Qia94b] have addressed the integrity related problems that arise when key-based functional and referential integrity requirements are enforced in multilevel relations (i.e., problems related to polyinstantiation). At the heart of their work is a model that differentiates between the data a user sees and data that a user believes. Qian calls these *accessibility* and *believability*, respectively. This distinction has been exploited to help resolve ambiguity in polyinstantiated relations.

However, it is not clear how this distinction can help solve the integrity related problems such as the one described in the introduction. It does not make much sense to have two tuples for each employee, one at High level that contains the correct salary and the other at the Low level with possibly an incorrect salary.

4 Formalization of Our Approach

Assume we are given a valid database state D of a multilevel secure database containing data at classification levels l_1, l_2, \dots, l_n . The classification level of a data item t is denoted by $L(t)$. Assume further that the data in this database must satisfy integrity constraints. An integrity constraint is an assertion (or a predicate) on the database state. A database state D is *valid* if all integrity constraints hold in D .

Given a valid database state D , an integrity constraint I , and a data item t , we are interested in determining if I holds in $D \cup t$. (For simplicity, we have chosen to drop $\{ \}$ around t .) We call t a prospective data item to be added to the database. Similarly, when a data item t must be deleted, we are interested in determining if I holds in $D - t$. (Modification of data items can be formalized similarly.)

In addition to the integrity constraints, we have a security policy which must be satisfied. The security policy specifies that the constraint must be evaluated without regard to data at higher classifications than the classification of the data to be inserted.

Note that the example in the introduction does not meet the security policy since data items classified High may determine if tuples classified Low are valid.

A constraint is said to be *level-valid* at level l if it can be verified as true or false using only data at level l or below. A constraint that is not level-valid is called a *multilevel-valid* constraint.

Suppose we have a constraint I that is multilevel-valid. To enforce both the constraint and the security policy, we wish to replace I by a collection of level-valid constraints

$$I' = \{I_{l_1}, I_{l_2}, \dots, I_{l_n}\}$$

such that

1. each I_{l_j} in I' is a level-valid constraint at level l_j , and
2. if D satisfies all level-valid constraints in I' , then D also satisfies I .

Hence replacement of constraints that are not level-valid by constraints that are level-valid satisfies both the integrity and the secrecy requirements.

Let \mathcal{V} denote the set of all valid database states under I , i.e., $\mathcal{V} = \{D : D \text{ satisfies } I\}$, and let \mathcal{V}' be the set of valid database states that satisfy the level-valid constraints in I' , i.e., $\mathcal{V}' = \{D : D \text{ satisfies } I'\}$.

If $\mathcal{V} = \mathcal{V}'$, then the original multilevel constraint has been exactly translated into a set of level-valid constraints. The more likely situation only guarantees that $\mathcal{V}' \subset \mathcal{V}$.

Since there are many possibilities for I' and, therefore, \mathcal{V}' , we may wish to compare the goodness of a replacement of an I by an I' . An example of an appropriate measure would be to compute the ratio

$$\frac{\text{card}(\mathcal{V}) - \text{card}(\mathcal{V}')}{\text{card}(\mathcal{V})}$$

Such a measure would become zero if all database states could be determined from data at or below the subject's level, and become one if no valid database states could be so determined.

5 Basic Idea

Example 1 To illustrate the basic idea, we consider once again the example given in the introduction. We replace I which is not level-valid by specifying two level-valid constraints as follows:

I_{High} : A High user would be allowed to insert a manager tuple if the new salary is not the lowest salary in the *MGR* relation.

I_{Low} : A Low user would be allowed to insert an employee tuple so long as the inserted salary is not the highest in the *EMP* relation.

With the assumption that the database originally satisfies the constraint I , after the addition of either a High or a Low tuple, the database will continue to satisfy the original constraint. Furthermore, both these constraints will permit insertions into *MGR* and *EMP* relations without checking data at another level. Thus, enforcement of these two level-valid constraints would guarantee that

1. a prospective tuple to be inserted by a High user carries no new information observable by a Low user, and
2. a prospective tuple to be inserted by a Low user retrieves no High information.

Note that there are two problems with our approach that we need to address. First, how do we insert tuples into an empty database state and, second, how do we insert tuples pertaining to the "lowest paid manager" or the "highest paid employee"? We will require trusted database administrator (DBA) processes to perform these operations since integrity checking will have to be suppressed during these operations. This is discussed more fully in Section 8.

Note that $I' = \{I_{\text{High}}, I_{\text{Low}}\}$ is not the only possible replacement for I . Indeed, I_{High} is needlessly restrictive since a High subject can read Low data without violating secrecy. Thus, a better possibility would be to use $I'' = \{I'_{\text{High}}, I_{\text{Low}}\}$ where

I'_{High} : A High user would be allowed to insert a manager tuple if the new salary is higher than the highest employee salary.

Like I_{High} , the new condition I'_{High} does not violate the secrecy requirement since High subjects are permitted to read Low data. However, I'_{High} is clearly superior to I_{High} since under I' only the tuples pertaining to the "highest paid employee" would have to be inserted by a trusted DBA

process. There are no restrictions on the "lowest paid manager" and these tuples can be inserted by untrusted subjects.

It is possible to further simplify the constraints so that only the single prospective tuple needs to be checked, not all the tuples in the Low relation. This is accomplished by specifying a fixed upper bound, say x , for the employee salaries. Obviously, this upper bound will become the lower bound for the manager salaries. The new level-valid integrity constraints are as follows:

I''_{High} : A High user would be allowed to insert a manager salary if the new salary is greater than x .

I''_{Low} : A Low user would be allowed to insert an employee salary so long as the inserted salary is less than or equal to x .

Although it is tricky to arrive at an appropriate value of x , the last set of integrity constraints has an additional benefit. The cost of checking if an insertion satisfies I''_{High} or I''_{Low} is much lower than that of any one of the preceding constraints. Indeed, [BBC80, BB81] advocates using this strategy to reduce the cost of enforcing integrity constraints, where it is also shown that a large class of constraints can be enforced using similar tactics. \square

Example 2 Suppose that we modify example 1 so that we have only one relation: $EMP(\text{name}, \text{salary}, \text{position})$ containing names, salaries, and positions of the employees. There are three positions, apprentice, manager, executive. The constraints are:

1. all apprentice salaries are less than any manager's salary, and
2. all manager salaries are less than any executive's salary.

The classification levels are: 1) apprentice records are classified confidential (C), 2) manager records are classified secret (S), and 3) executive records are classified top secret (TS).

Subjects at one level are not allowed to read information at a higher level, nor are they allowed to write at any other level.

Whenever a subject at the apprentice level attempts to write a new (apprentice) tuple with a salary, the database must decide if this is allowed. As noted above, if the multilevel-valid constraint is initially satisfied, then the following level-valid constraint:

I_C : New apprentice salary must be less than the highest existing apprentice salary.

is sufficient to guarantee that a new tuple does not invalidate the original constraint.

The highest cleared category, executive, needs only to be concerned with inserting salaries higher than the highest paid manager. This is similar to the relation between manager and employee in preceding example and is accomplished by requiring the following level-valid constraint:

I_{TS} : New executive salary must be higher than the highest existing manager salary.

which would guarantee that any executive salary meets the original multilevel-valid requirement.

Finally, if a user attempts to insert a new manager tuple there are two checks must be performed: the salary value must be higher than any apprentice, but lower than any executive. The constraint

New manager salary must be less than the highest existing manager salary.

is not sufficient since it is possible that the new manager salary is lower than the highest paid apprentice. Since this is also not allowed, an additional constraint must be added to the preceding constraint as follows:

I_S : New manager salary must be less than the highest existing manager salary, but greater than the highest paid apprentice.

which is sufficient. □

Thus, in general, there are several ways to replace a multilevel-valid constraint by a collection of level-valid constraint. A wise replacement will not only accurately reflect the policy with respect to secrecy and integrity, but minimize the number of valid database states that are attainable from updates by trusted subjects only.

The previous examples have illustrated integrity constraints which are affected only by insert operations. The following example illustrates integrity constraints which are affected by delete operations also. It is important to note that constraints that are enforced during an insert operation may be different from the constraints that are enforced during a delete operation.

Example 3 Suppose that we modify the relation of example 2 by adding a new column, called 'proj#', recording for each employee the project the employee is working on. Thus, the relation EMP has the following schema $EMP(name, salary, position, proj\#)$. Classification levels of tuples in the extended EMP relation are the same as in example 2.

Suppose that the following constraint must be enforced:

The average salary for employees working on project P200 must be greater than 5000.

Suppose that a subject at the apprentice level attempts to delete a tuple of an apprentice working on project P200. Since employees in any position can work in project P200, evaluation of this multilevel-valid constraint would require evaluating the average on data items that are classified at confidential levels as well as at higher levels. Note, however, that, if this multilevel-valid constraint is initially satisfied, the following level-valid constraint

I_C : The *new* average salary over all apprentice tuples with project value = 'P200' (i.e., not including the salary of the tuple to be deleted) must be greater or equal to the *old* average salary over all apprentices tuples with project value = 'P200' (i.e., including the salary of the tuple to be deleted).

is sufficient to guarantee that the delete operation does not invalidate the multilevel-valid constraint.

Similarly, if a subject at secret level tries to delete a manager, the following level-valid constraint would be sufficient to ensure the validity of the original multilevel-valid constraint:

I_S : The *new* average salary over all apprentice and manager tuples with project value = 'P200' (i.e., not including the salary of the manager tuple to be deleted) must be greater or equal to the *old* average salary over all apprentices and manager tuples with project value = 'P200' (i.e., including the manager salary of the tuple to be deleted).

Note that although constraint I_S involves tuples from two levels, it is still a level-valid constraint since it can be evaluated over tuples that are classified either secret or confidential.

6 Theoretical Basis

6.1 General approach

The discussion to this point has been either very general or concerned with a specific example. We will now show how these approaches are related. That is, we will develop a method of translating

multilevel- valid constraints into a set of level-valid constraints. We will develop the method considering the features of a conventional (single-level) DBMS. As this is done, however, we will distill the critical features and generalize the method to apply to more arbitrary functions.

6.2 Notation

The first necessary step that is to define a formalism for specifying constraints. Constraints are generally formed by comparing characteristics of two sets of data (i.e. employee salaries and manager salaries). This can be formalized in the following definition.

Definition 1 A constraint is an expression of the form

$$f_1(q_1(R)) \Theta f_2(q_2(R))$$

where f_1, f_2 are functions resulting in a numerical value or a set of specific string values; q_1, q_2 are queries which operate on the relation R to produce a restricted relation or view; and Θ represents one of the standard comparators: $>$; $<$; \leq ; \geq ; $=$; or the existential operator "exists in". The constraint will be true for any valid database state.

In later discussions, Ω will denote Θ or $=$, so if $\Theta = <$, then $\Omega = \leq$.

Example 4 In our notation, the constraint "all apprentice salaries are less than any manager's salary" becomes:

$\text{MAX}(\text{select } salary \text{ from } EMP \text{ where } position = \text{'apprentice'}) < \text{MIN}(\text{select } salary \text{ from } EMP \text{ where } position = \text{'manager'})$.

□

This formalism allows sophisticated, complex relationships to be expressed. The views q_1, q_2 are general and the functions f_1, f_2 are not limited to those predefined by the system. In fact, the functions could be arbitrary procedures implemented by triggers. For simplicity, however, we will limit our discussions to the standard database aggregate functions, SUM, COUNT, MAX, and MIN.

Note that this formalism incorporates much of the characterization of constraints that are usually expressed in a language which is like relational calculus [Sto75, BBC80, BB81, BM88, GW93] but in a different notation. As an example, [BM88] uses a tuple-based notation utilizing a precondition (the selection criteria for data to be evaluated), and condition (the evaluation function itself), as well as the aforementioned aggregate functions. Although both systems are capable of expressing the required constraints, our method does not require the new terminology and notation found in [BM88]. For example we can allow aggregate values as f_1, f_2 in our notation; a precondition which we implement as views q_1, q_2 ; and a test condition which we implement as Θ .

Gupta and Widom [GW93] approach also uses distinctly different notation, namely first order logic (which can, of course, be translated into more familiar SQL) and is stated as a condition for failure to meet the constraint. Their notation requires that the selection/projection clauses (implemented as views in our notation) be combined into a single formula. Testing is then done only for existence conditions, the aggregate functions are not allowed. Such existence conditions are not sufficient to test for the aggregate functions COUNT or SUM, hence our proposed notation is more flexible and allows us to address additional constraints. The function and view based notation presented here is based upon the familiar SQL and is much more intuitive than any of the previously proposed notations. It maintains separation of important concepts, allows for multiple aggregate

functions and is not restricted to using base relations or conjunctions of simple selects as are the previous studies.

To illustrate how constraints are specified in our proposed format, we consider the following example taken from [BM88].

Example 5 Let us consider a database consisting of the following relations:

EMPLOYEE(*emp#*, *name*, *salary*, *address*, *proj#*, *dno*)

PROJECT(*proj#*, *name*, *mgr#*, *budget*, *location*)

MANAGER(*mgr#*, *name*, *age*, *salary*, *address*)

The following integrity constraints may be defined on these relations:

- I_1 : The project budget must be greater or equal to zero.
- I_2 : For projects located in Italy there can be at most two managers.
- I_3 : The average salary for the employees working on project 'P200' must be greater than \$5.000.
- I_4 : The sum of the salaries for the employees working on a project must be less than the project budget.
- I_5 : Each employee must work in an existing project.

In our notation, these constraints can be expressed as follows:

- I_1 : p_i "exists in" {select *proj#* from *PROJECT*} AND {select *budget* from *PROJECT* where *proj#* = p_i } = 0.
- I_2 : COUNT{select *mgr#* from *PROJECT* where *location* = 'Italy'} \leq 2.
- I_3 : AVE{select *salary* from *EMPLOYEE* where *proj#* = 'P200'} > \$5.000.
- I_4 : p_i "exists in" {select *proj#* from *PROJECT*} AND SUM{select *salary* from *EMPLOYEE* where *proj#* = p_i } < {select *budget* from *PROJECT* where *proj#* = p_i }
- I_5 : e_i "exists in" {select *emp#* from *EMPLOYEE*} AND {select *proj#* from *EMPLOYEE* where *emp#* = e_i } "exists in" {select *proj#* from *PROJECT*}

□

One difficulty, however, that is found in this notation but addressed in both [BM88] and [GW93], is expressing constraints that hold, not on the set, but on each member of some set. This is handled by formulating two constraints, connected by a conjunction, as illustrated by I_1 , I_4 , I_5 above.

This extended form of constraint is formalized by the following definition.

Definition 2 A complex constraint is an expression of the form

IC_i AND IC_j

where either both IC_i and IC_j are simple constraints as defined in Definition 1, or IC_i is a simple constraint and IC_j is a complex constraint.

6.3 Ordering

The critical feature of this notation is the fact that if both sides of an integrity constraint yield numerical values, it provides us with a way to order the relations using the comparator Θ (in cases where either $f_1(R)$ or $f_2(R)$ yields string values, Θ must be “=” or “exists in”). The fact that the relations are ordered allow us to derive some simple tests for determining if a tuple may be added to (or deleted from, or modified in) the database.

The first necessary observation is that Θ is transitive.

Lemma 1 Θ is transitive. That is, if $A \Theta B$ and $B \Theta C$, then $A \Theta C$. More generally, if either (i) $A \Omega B$ and $B \Theta C$ or (ii) $A \Theta B$ and $B \Omega C$ holds, then $A \Theta C$ holds.

The following lemma gives two tests for a prospective tuple, which, if both are satisfied, guarantee the continued satisfaction of an existing constraint.

Lemma 2 Given a constraint $f_1(q_1(R)) \Theta f_2(q_2(R))$, which is known to be satisfied by the current state of the database, then if there exists a tuple t satisfying the following two conditions:

1. $f_1(q_1(R \cup t)) \Omega f_1(q_1(R))$, and
2. $f_2(q_2(R)) \Omega f_2(q_2(R \cup t))$,

then the database will still satisfy the constraint after t is added.

Proof: Since $f_1(q_1(R \cup t)) \Omega f_1(q_1(R))$ and $f_1(q_1(R)) \Theta f_2(q_2(R))$, from the Lemma 1 it follows that $f_1(q_1(R \cup t)) \Omega f_2(q_2(R))$. This last expression when combined with $f_2(q_2(R)) \Omega f_2(q_2(R \cup t))$ yields $f_1(q_1(R \cup t)) \Theta f_2(q_2(R \cup t))$, which is a statement that the database, after the addition of tuple t , satisfies the constraint. \square

The preceding lemma specifies two conditions, one for each expression in the constraint. We can therefore define two sets of valid tuples, one for each condition. Those tuples in both sets may be added to the relation and still satisfy the original constraint.

Fortunately, in many cases of practical interest, a substantial number of tuples are in both sets. It is even common for a tuple to influence only one of the conditions in Lemma 2. For example, if $f_2(R)$ is equal to a constant, then it is true that $f_2(q_2(R)) = f_2(q_2(R \cup t))$, regardless of what tuple t is added to the database. In such cases, tuples only influence one condition, so the conjunction does not present a serious problem. The constraint in example 4 illustrates one such constraint.

Example 6 Consider the constraint given in example 4. From Lemma 2, it follows that we need to satisfy the following two conditions:

1. $\text{MAX}\{\text{select salary from EMP} \cup t \text{ where position} = \text{'apprentice'}\} \leq \text{MAX}\{\text{select salary from EMP where position} = \text{'apprentice'}\}$, and
2. $\text{MIN}\{\text{select salary from EMP where position} = \text{'manager'}\} < \text{MIN}\{\text{select salary from EMP} \cup t \text{ where position} = \text{'manager'}\}$

Apprentice tuples with salaries less than or equal to the present maximum will satisfy (1), indeed, they will not change it. All apprentice tuples will also satisfy (2) since it is unchanged by their addition, regardless of their salary. The addition of an apprentice tuple therefore only requires evaluation of one level-valid constraint:

I_C : If $t.position = \text{'apprentice'}$, then $t.salary \leq \text{MAX}\{\text{select salary from EMP where position} = \text{'apprentice'}\}$.

□

For those situations where all tuples with some characteristic (such as $position = \text{'apprentice'}$) satisfy one of the lemma 2 conditions, regardless of other characteristics, only one view needs to be considered. This allows the constraint to be simplified for performance reasons even without considering classification constraints. As [GW93] pointed out, such performance benefits are especially valuable in distributed databases. If the apprentice and manager tuples are kept at distinct physical locations, the constraint can still be verified without requiring a distributed join.

Example 7 Consider the relation from example 2 with the new constraint “all apprentice salaries must be less than the average salary,” i.e., $\text{MAX}\{\text{select salary from EMP where position} = \text{'apprentice'}\} \leq \text{AVE}\{\text{select salary from EMP}\}$.

Here the conditions from lemma 2 are:

$\text{MAX}\{\text{select salary from EMP} \cup t \text{ where position} = \text{'apprentice'}\} \leq \text{MAX}\{\text{select salary from EMP where position} = \text{'apprentice'}\}$ AND $\text{AVE}\{\text{select salary from EMP}\} \leq \text{AVE}\{\text{select salary from EMP} \cup t\}$.

Here each new tuple affects both conditions. Apprentice tuples satisfy the first condition if the new salary is less than the existing maximum apprentice salary. The second condition specifies that the new salary must be greater than the existing average over all the tuples. Note that the addition of a very low apprentice salary may reduce the average salary below an already existing apprentice salary. The range of salaries satisfying both conditions may be very small. □

A lemma, similar to Lemma 2, holds for the delete operation.

Lemma 3 Given a constraint $f_1(q_1(R)) \Theta f_2(q_2(R))$, which is known to be satisfied by the current state of the database, then if there exists a tuple t satisfying the following two conditions:

1. $f_1(q_1(R - t)) \Omega f_1(q_1(R))$, and
2. $f_2(q_2(R)) \Omega f_2(q_2(R - t))$,

then the database will still satisfy the constraint after t is deleted.

The proof of the above lemma is similar to that of Lemma 2 and thus we omit it.

7 Level-valid Constraints

The preceding discussion would be sufficient to simplify constraints to a single view if all the data was classified at a single level. The functions and views used at a specific level need not be the same as those used in the multilevel-valid constraint. They do not even have to bound the multilevel functions, but they must increase or decrease appropriately in order to maintain the relationship (Θ) in the multilevel-valid constraint. For example, consider the level-valid constraints I''_{High} and I''_{Low} in Example 1. These level-valid constraints do not mention a range at all; however, these replacements work since they guarantee continued satisfaction of the multilevel-valid constraint. This is made more precise in the following theorem.

For this theorem, we define functions f_3 and f_4 and views q_3 and q_4 operating only on data at level l or below that allow us to express conditions satisfying the original multilevel-valid constraint. The notation R_l is used to denote the subset of data in relation R classified at or below level l .

Theorem 1 Given a multilevel-valid constraint $f_1(q_1(R)) \Theta f_2(q_2(R))$, which is satisfied by the current state of the database R , if there exist functions f_3, f_4 and views q_3, q_4 such that

- (a) $f_1(q_1(R \cup t)) - f_1(q_1(R)) \Omega f_3(q_3(R_l \cup t)) - f_3(q_3(R_l))$, and
- (b) $f_2(q_2(R)) - f_2(q_2(R \cup t)) \Omega f_4(q_4(R_l)) - f_4(q_4(R_l \cup t))$

hold for all tuples t . In addition, for tuples t_l at level l or below,

- (c) $f_3(q_3(R_l \cup t_l)) \Omega f_3(q_3(R_l))$, and
- (d) $f_4(q_4(R_l)) \Omega f_4(q_4(R_l \cup t_l))$.

Then $f_1(q_1(R \cup t_l)) \Theta f_2(q_2(R \cup t_l))$. That is, the constraint is still satisfied after the tuple t is added.

Proof: We only consider the case when Ω produces a numeric value (other cases follow similarly). Rearranging (a) gives $f_1(q_1(R \cup t)) \Omega f_1(q_1(R)) - (f_3(q_3(R_l)) - f_3(q_3(R_l \cup t)))$. Since $O \Omega f_3(q_3(R_l)) - f_3(q_3(R_l \cup t_l))$ from (c), adding these gives $f_1(q_1(R \cup t_l)) \Theta f_1(q_1(R))$. A similar argument, utilizing (b) and (d) yields $f_2(q_2(R)) \Omega f_2(q_2(R \cup t_l))$. So, by lemma 2, the constraint is still satisfied after adding the new tuple. \square

Once the functions f_3, f_4 and views q_3, q_4 are found, the constraint may be checked without reference to High classified information. To the Low user, the constraints appear to be conditions (c) and (d) of the theorem. Conditions (a) and (b) are used in the design stage to find suitable functions and views, but are not visible to Low users. While the relation R_l must be restricted to contain only data at level l or below, the tuple t , being added to the database, need not be further restricted since it already contains only data at level l or below.

To simplify the functions, we will usually choose $f_1 = f_3, f_2 = f_4, q_1 = q_3$, and $q_2 = q_4$, but this is not required.

A similar theorem holds with respect to delete operation.

8 Reaching Database States that do not satisfy Level-valid Constraints

It is still somewhat unclear how the initial consistent database state is to be reached, and how to reach those valid database states that fall outside the level-valid constraints (i.e., database states that are in $\mathcal{V} - \mathcal{V}'$). That is, we have developed a technique that allows us to reach many database states without referring to High data, but we must have some technique allowing us to reach all database states.

These additional techniques will require checking High data to ensure that the original multilevel-valid constraint is still satisfied, even if the level-valid one is not. Such procedures must therefore be trusted. However, being trusted may not be enough, since the database has now implemented the more specific constraints, which must be bypassed. Trusted subjects have the authority to downgrade information, but may not have authority to actually bypass general database integrity restrictions. Generally a DBMS gives the ability to deal with constraints only to the DBA. The additional database states can therefore only be reached through a trusted DBA.

8.1 Initial state

If a constraint depends upon data in the database, as we are considering, how are the initial tuples loaded? Constraints such as we are considering require a substantial amount of data in the database

in order to be evaluated properly. Otherwise, many legitimate tuples will require special DBA treatment as they extend the limits of the datasets used in the constraints. In several commercial systems, the database may be initialized using a COPY command, loading large quantities of data. During the initial loading, the constraints are disabled, that is, the data is assumed to already be verified as meeting these constraints. This procedure is useful if valid data already exist in some other DBMS or file management scheme. If valid data does not exist, it may be necessary to estimate the values in the constraints using fixed values instead of values derived from views (see the last set of level-valid constraints in Example 1). After inserting some of the data the constraint could be changed to a form dependent upon a view of that data.

8.2 Remaining States

Now suppose that a tuple has failed to meet the constraint at its level, but needs to be inserted anyway? We can assume that some user has been granted both trusted status and DBA privileges. This trusted DBA must then disable the constraint at the tuple's level, and disable other inserts at that level, while they insert the new tuple. Some products (e.g., Oracle) provides this capability, so the DBA can disable the individual constraint while the tuple is inserted. A similar capability to disable triggers is desirable when dealing with systems utilizing that method of implementation (i.e. Sybase). Some systems, such as Ingres, only provide the capability to disable all constraints, not single ones. An alternative in this case would be for the DBA to 1) delete the rule, insert the tuple and re-insert the rule or 2) suspend all rules, insert the tuple and reactivate the rules. Such problems indicate the importance of minimizing the number of times that trusted processes must be used to insert tuples. The alternative to using the level-valid constraints as developed here is to perform this process for every addition, not just those that are not allowed by level-valid constraints.

9 Conclusions

We have shown that there is a large class of multilevel-valid integrity constraints that can be transformed into multiple level-valid constraints whose satisfaction is sufficient to ensure that the original multilevel-valid constraint is also satisfied. The level-valid constraints, by definition, are free from signaling channels. The price for this is that certain modifications that are valid under the original constraint, may not be valid under the level-valid constraints. It is possible make such modifications if we rely on trusted processes to do so.

As part of our current work, we are investigating methods that automatically generate a suitable set of level-valid constraints for certain multilevel-valid constraints such as aggregation. We are also investigating how given a set of integrity constraints, triggers may be automatically generated for the support and repair of integrity constraints in a secure way. By repair it is meant that some actions are executed to restore the database correctness with respect to the violated integrity constraint.

References

- [AD87] S. G. Akl and D. E. Denning. Checking classification constraints for consistency and completeness. In *Proc. of the IEEE Symp. Security and Privacy*, pages 196–201, 1987.
- [BB81] Philip A. Bernstein and Barbara T. Blaustein. A simplification algorithm for integrity assertions and concrete views. In *Proc. of IEEE Int'l. Computer Software & Applications Conf.*, pages 90–99, 1981.

- [BBC80] Philip A. Bernstein, Barbara T. Blaustein, and Edmund M. Clarke. Fast maintenance of semantic integrity assertions using redundant aggregate data. In *Proc. of Int. Conf. on Very Large Data Bases*, pages 126–136, October 1980.
- [BM88] E. Bertino and D. Musto. Correctness of semantic integrity checking in database management systems. *Acta Informatica*, 26:25–57, 1988.
- [Bur90] Rae K. Burns. Integrity and secrecy: Fundamental conflicts in database environment. In *Proc. 3rd RADC Database Security Workshop*, pages 37–40, June 1990.
- [Den86] Dorothy E. Denning. A preliminary note on the inference problem in multilevel secure database systems. In *Proc. National Computer Security Center Workshop on Database Security*, June 1986.
- [GW93] Ashish Gupta and Jennifer Widom. Local verification of global integrity constraints in distributed databases. In *Proc. of the ACM SIGMOD Int'l. Conf. on Management of Data*, pages 49–58, 1993.
- [MJ88] Catherine Meadows and Sushil Jajodia. Integrity versus security in multi-level secure databases. In Carl E. Landwehr, editor, *Database Security, Status and Prospects*, pages 89–101, Amsterdam, 1988. North-Holland.
- [Qia94a] Xiaolei Qian. Inference channel-free integrity constraints in multilevel relational databases. In *Proc. IEEE Symposium on Security and Privacy*, pages 158–167, May 1994.
- [Qia94b] Xiaolei Qian. A model-theoretic semantics of the multilevel relational model. In *Lecture Notes in Computer Science*, pages 201–214, Berlin, 1994. Springer-Verlag.
- [Sto75] Michael Stonebraker. Implementation of integrity constraints and views by query modification. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 65–78, 1975.
- [SW92] Ken Smith and Marianne Winslett. Entity modeling in the MLS relational model. In *Proc. of Int. Conf. on Very Large Data Bases*, pages 199–210, 1992.
- [Thu91] B. M. Thuraisingham. A nonmonotonic typed multilevel logic for multilevel secure database/knowledge-base management systems. In *Proc. IEEE Workshop on Computer Security Foundations*, pages 127–138, 1991.

TOP: A Practical Trusted ODBMS

Marvin Schaefer^a

Valerie A. Lyons^b Paul A. Martel^b Antoun Kanawati^b

^aArca Systems, Inc., 10320 Little Patuxent Parkway, Suite 1005, Columbia, MD 21044-3312, USA

^bONTOS, Inc., Three Burlington Woods, Burlington, MA 01803, USA.

Abstract

The Trusted ONTOS Prototype (TOP) is a new research initiative into secure object database management. TOP presents a set of features from object data management, C++ application development, confidentiality, and integrity to provide the balance of usability, safety, and flexibility that ONTOS users have learned to enjoy over the last decade. Although several paper studies have been conducted by others, we and our sponsors have concluded that the time is right to produce a prototype that can be used as a testbed. TOP should help to assess confidentiality/integrity tradeoffs, efficacy, and performance issues. In addition, it will be possible to gauge the relative size and complexity of TOP's client/server TCB. This paper presents an overview of TOP's access control policy, features, and philosophy.

1. INTRODUCTION

Objects are everywhere. The need has surfaced for trusted systems. To move with the trend toward object technology, the passive object paradigm of relational database management has also shifted to the look and feel of genuine object databases (ODBMS). However, active objects present access control challenges different from those addressed under the Bell-LaPadula model. In studies produced to date, the exclusive focus on confidentiality has generally eschewed the spirit of object data management. From these studies [2,3,4,7] it became clear that the emphasis on finding a high-assurance mechanism to control access has begun to overshadow the goal of providing useful multilevel access to object databases. ONTOS, as a principal vendor of ODBMS, has recognized the need to conduct security research¹ in the context of contemporary object technology.

1.1 Objectives: A Usable B1 Client/Server ODBMS

As a research effort, the TOP developers were given the opportunity to choose a set of features from object data management [8], C++ application development, confidentiality, and integrity to provide the balance of usability, safety, and flexibility that ONTOS users have learned to enjoy.

Defining assurance for a B1 object data management system has been a continuing challenge during the project. Neither the TCSEC nor the TDI sheds direct light on either the security issues or acceptable means of their resolution. Over the last five years, there has been increasingly active research into defining access control in ODBMS. Readers of the current literature easily become aware of lack of consensus among the researchers on many of the most fundamental issues (*e.g.*, inheritance, the object model itself). Many trusted

¹This project is funded by Rome Laboratory under Contract No. F30602-93-C-0123

ODBMS researchers have worked from the perspective of the relational model, which has had the undesirable effect of missing the major issues of the common object programming models (*C++*, *Smalltalk*). TOP explores the implications of a *de facto* object model (*C++*) on security concerns, and attempts to resolve conflicts between the model and confidentiality requirements².

The B1 level was chosen deliberately to provide the TOP team with the freedom to experiment in the design of a first-time-ever TCB without having to be concerned with: TCB minimality, least privilege, least common mechanism, or covert channels, as would be required for higher TCSEC levels. We believe that this is proper; the lack of worked examples provides us with no advance insight into either the challenges that will be encountered in a real system, or into the tradeoffs needed for their resolution. We have concluded that the prototype, once implemented and used, will give us valuable information about the potential for achieving B2 or higher assurance in trusted ODBMS architectures.

We and our sponsors decided that we would benefit more by undertaking a proof-of-concept prototype development than we would by conducting a theoretical study. We intend our results to provide new knowledge in the theory and design of trusted ODBMSs, and an empirical validation of such theories and designs.

Designing and maintaining a good database has become an art form in and of itself. The subtleties of security policy, particularly arcane concepts like the \star -property and its consequences often serve to befuddle users and through confusion make things harder than they need to be. TOP strives to minimize constraints and to encourage natural interaction between user and application. The philosophy in TOP has not been the dismissal of prudent confidentiality concerns in favor of database integrity or vice-versa. It has instead been to recognize that the reason for the existence of the \star -property is to control untrusted, potentially malicious, application code. We believe this can only be done by shutting out untrusted code at critical junctures where confidentiality and integrity objectives are in potential conflict. This is achieved through a private multilevel dialogue between just the cleared user and the TCB³.

Here are our priorities:

- **A well thought-out marriage of confidentiality and integrity.** Everyone knows what is meant by confidentiality; by 'integrity' we mean that the user be provided with tools and functionality to produce and maintain a consistent representation by an object database of some aspect of the real world. This means that the user need be provided the ability to get it right, to keep it right, and to know if it is right.
- **A sound means of resolving conflicts between confidentiality and integrity objectives.** Since the best knowledge of the relationships between the data and the real world resides with the user, we believe that the resolution of confidentiality/integrity conflicts can best be resolved by trusted dialogue between the informed, cleared user and the TCB. We propose an environment where the DBA and the cleared

²For example; under confidentiality requirements, the class graph (schema) need not be fully visible at all levels; a coherent multilevel object model, on the other hand, would seem to require that the subclass relations of the class graph be identical at all levels. In TOP, the class graphs of each visibility level are not necessarily isomorphic to each other (compromise towards security), and multiple inheritance is not supported due to the complications it introduces into multilevel type analysis, and the definition and implementation of uniqueness concerns. Also, the choice of *C++* has forced TOP to address the daily issues of programming: multilevel source code (*C++* header files are the schema descriptions), program object code is not stored in the database and methods bodies may vary among the different levels of an object.

³Security policies include *authorized users* to reconcile those conflicts that cannot be handled through an automated policy; for example: downgrading data, simultaneous cooperating object updates at multiple levels, and any other apparent violation of the \star -property. Because of the need for an isolated trusted path that can be invoked during a session by either the *user* or by the TCB, TOP assumes the existence of a B3 trusted path.

users cooperate to resolve data conflicts, and provide true multilevel transactions for that purpose.

- **User views that accommodate clearance and need-to-know.** Objects and relationships should be presented to users consistent with the security profile of the application environment. Views, including multilevel views, can provide the application with timely presentations of data that are suitable and safe for presentation to untrusted code.
- **Cover stories, not accidental polyinstantiation.** The database should represent reality except in those cases where an 'alternate reality' is required by sufficiently cleared users. The confidentiality policy ought not to be allowed to corrupt a database solely because of the over-restrictiveness of the ★-property.
- **Truly object-based, not relational-based, policy model and implementation.** Work done by many previous researchers has extrapolated from the well-known relational security models. We, on the other hand, have revisited the implications of the model from the perspective of the object-oriented environment, to produce a fresh outlook for ODBMS security. Furthermore, the choice of C++ is now becoming even more widely accepted as an application programming interface, and our implementation will be as faithful to this regime as is practicable.

1.2 Overview

In the remainder of this paper we present the access control policy, beginning with an overview and progressing into details of the operations of creation, viewing, modification and deleting multilevel objects. An example is provided to illustrate novel features of our work. The paper concludes with a description of the project plan and status.

2. POLICY OVERVIEW

TOP is intended to satisfy the mandatory and discretionary policy requirements for B1, as specified in the TCSEC. TOP bases its mediation on a combination of factors that include: the user's clearance, the environment from which the user logs in, the user's login level, and the security attributes of TOP objects being accessed. In principle, TOP objects may all be multilevel, and may also be subject to discretionary access controls.

2.1 MAC: View of Level-Dominated Data

TOP Users are represented by untrusted clients that are labeled with the user's login level. The client is provided with a *view* of TOP objects (*viz.* schema, properties and procedures). The view is derived from single level 'slices' of TOP objects (*i.e.*, the *l*-instantiations) whose level, *l*, is dominated by the client's level. Such views are consistent with the simple security and discretionary security conditions of Bell-LaPadula.

For the time being, TOP's policy model is restricted to hierarchical levels and does not encompass the complete lattice model. This is because of the evolving semantics of the *l*-view as an alternative to automatic polyinstantiation. The rationale behind our postponing a generalized lattice is given in Section 3.3, below, where we describe *scooping* and its uses. Complications also arise because TOP allows schema-level cover stories⁴.

2.1.1 Update at Level

TOP treats all clients as untrusted subjects. In normal operation, this is quite adequate for retrieving and updating data. Creation and modification of data is performed at the level of

⁴Assume the reader has permissions $\langle top, \{A,B\} \rangle$, and is reading an object *X*. Assume that *X* is instantiated only at $\langle confidential, \{A\} \rangle$ and $\langle confidential, \{B\} \rangle$. The user may view both instantiations under the mandatory access control policy; however, it is possible that the schema corresponding to the two compartments may be different. Thus, the state and the precise type of the object can be ambiguous when compartments are used.

the client, and is consistent with the ★-property and discretionary security conditions of Bell-LaPadula. TOP stores the data persistently as a faithful image of the client's view from its level. Unlike trusted RDBMSs, untrusted updates of this form *do not* cause polyinstantiation. Accidental polyinstantiation occurs in trusted RDBMSs as a consequence of the ★-property and multilevel operation. In fact, as we show below, TOP's semantics precludes the possibility for accidental polyinstantiation to occur.

2.1.2 *Trusted Updates to Strictly Dominated Levels (Downgrades)*

TOP supports *multilevel* updates through a trusted dialogue between an authorized, appropriately cleared user and the TOP TCB. The classes of update supported are: downgrade, update of multiple levels of data at the same time, and cover story (intentional polyinstantiation). The mechanism used to support the dialogue is a fully-isolated B3 Trusted Path that is invoked either by the user or by the TOP TCB.

2.2 Discretionary Access Control

We plan for TOP to provide DAC controls to support rôles for DBA and SSO along with mode-based access controls to the granularity of individual users and groups. At present there is no agreement within the community over issues of the scope of DAC in object contexts and over acceptable mechanisms for its assured implementation. We plan to design a DAC mechanism to provide DAC over named objects, including databases, object instantiations, and procedures.

2.3 Integrity: DBA-Provided Base Types

Most customers consider the preservation of database integrity a higher priority than control over disclosure. Integrity determines the continuing utility of the data and its interrelationships. Database operations are traditionally based on the notion of transactions. A transaction is a set of operations that read and/or write persistent objects and satisfies the *ACID* properties (atomicity, consistency, isolation, and durability). Briefly, *atomicity* means that the transaction is either executed in its entirety or not executed at all; *consistency* means that the transaction maps a database from one consistent state to another; *isolation* means that the transaction does not read intermediate results of other non-committed transactions; and, *durability* means that once a transaction is committed, its effects are guaranteed to endure despite system failures. Scheduling of transactions, *i.e.*, locking of data, needs to be accomplished such that the user application is notified of the success or failure of each transaction. This notification, unfortunately, could lead to illegal information flows and conflict with confidentiality policy requirements.

Modern DBMSs implement DBA-defined integrity checks that dynamically monitor data creation and update. Since the use of bad data will propagate additional bad data, we want to ensure that each transaction reflects the user's intention at the time of the commit.

In multilevel data management there is a potential for conflict between confidentiality and integrity concerns. This conflict comes about as follows: In the presence of multilevel integrity constraints, updates to low data may be constrained by more sensitive values. If the multilevel integrity constraint is known at low, it may be possible for an interloper to derive or infer specific information at high through probing. Existing literature has identified multilevel integrity problems regarding conflicting data values; referencing non-existing values; deleting referenced values; and, accidental or policy-induced polyinstantiation of data. TOP is designed to address these problems as well as others.

2.3.1 *Multilevel Integrity Constraints*

Triggers are one way in which ONTOS supports single-level integrity constraints. Because of TOP's support for multilevel objects, even single-level integrity constraints may involve a comparison of the multilevel values that had been instantiated. Validation of an update, therefore, may run afoul of an integrity constraint, independent of the level at which the client is acting. Notification of the client would not be permissible if some of the con-

strained data were classified at a higher level than the client. In such cases, TOP invokes the B3 Trusted Path to notify an appropriately cleared person of the violation. The person may well be the user on behalf of whom the client is operating. She would be notified if her workstation were located in an sufficiently cleared area, and if she were cleared sufficiently to receive notification. Such notification would be safe, since the Trusted Path mechanism is isolated from untrusted domains and the communication is between a cleared human and the TOP TCB. If it is not possible to notify the user directly, TOP will notify the DBA. The cleared user may then take appropriate action with respect to the update as a multilevel transaction.

2.3.2 *Polyinstantiation Control Specific to Object Instantiation*

Trusted RDBMSs can get badly corrupted from rigid adherence to the ★-property: an untrusted subject is forced to write data at its own level, and cannot even lock data at lower levels [8]. In contemporary trusted RDBMS architectures, updates to existing low data cannot be performed from a higher level without the possibility of a Trojan Horse compromise, since there is no assured means for the user to verify that *only the intended* information flow transfers from high to low. The only viable workaround has been to log out of the higher level and perform the update after logging back in at the lower level. This jeopardizes the ACID properties of the transaction: since all the data in a multilevel transaction cannot be not locked during a manually-implemented⁵ multilevel transaction, data can become damaged by other users' transactions and conversely.

When accidental polyinstantiation does occur, there can be far-reaching deleterious effects. For example, aggregate functions would likely return unpredictable values. These in turn, can, if used as the basis for decisions or future updates, imperil the correctness of the entire database.

But, sometimes polyinstantiation is necessary, although this is only when the intention is to deceive. In that case, the deception is called a 'cover story', the truth is instantiated at a higher level⁶, and the user needs to be able to accomplish this operation without sacrificing any of the ACID properties.

As shown below, TOP addresses this problem directly.

3. INSIDE TOP

The TOP access control policy model [9] is derived from the Bell-LaPadula family of models. In the initial prototype, however, we have chosen not to deal with the full lattice of compartments and are using only hierarchical levels. This has been done to simplify definitions needed for deriving views of multilevel objects. This allows us to uniquely identify *the nearest level* dominated by a specified level and eliminate complications caused by dominated non-comparable compartments. The section on views shows a motivation for this simplifying decision. Future treatments of TOP will address the general case.

3.1 Multilevel Schema and Property Visibility Levels

Every TOP object is derived from a type specified in the database schema. Types contain a set of properties and procedures (attributes). These are either explicitly defined in the type, or inherited from a supertype. A type is a subtype when it inherits attributes from a parent type, which is called the supertype. Inheritance is a means of organizing database types into a meaningful framework. Each of the attributes is assigned an explicit *visibility level* below which it is invisible. A sensitivity label that dominates the visibility level is assigned

⁵It would appear to a system as though distinct single-level untrusted logical subjects were independently involved in separate transactions. An alternative, such as that in replicated RDBMS architectures like that of SINTRA places responsibility for the complete multilevel transaction on multiple untrusted single-level DBMS servers.

⁶Readers of spy stories may be aware of onion-like layers of cover stories upon cover stories.

at the point of instantiation rather than globally over the type so as to afford maximum flexibility and control over information access. In order to provide multilevel views of schema and objects, we have chosen to define the visibility level of the type to be equal to the greatest lower bound of its non-inherited attributes.

Inherited attributes are assigned an induced visibility level equal to the visibility level of the type or of the attribute, whichever is higher. TOP supports multilevel inheritance. The visibility of the inherited attributes is governed by a rule that is explained following the discussion on *l*-instantiations.

Multiple inheritance, though supported in C++, is not supported by TOP.

3.2 Example

An example will be presented here in order to illustrate various issues on multilevel schema, views, polyinstantiation, and deletion. A typical scenario involving this database is that once a drug has finished its testing phase and has been released, the information regarding the test results has varying degrees of sensitivity. The label on the drug is a kind of cover story, while the actual test results (possibly indicating adverse effects) may need to be classified as top secret. The number of lawsuits involving this drug may direct the need to desensitize some of the information, or at least update the counter_indications labeling.

In this example, we use a hypothetical schema from a database at a pharmaceutical company. The part of the schema we will look at consists of two class definitions: Drug and Status. The table below shows the properties of these two classes, along with the sensitivity labeling for each property.

Drug			Status		
	C++	Name		C++	Name
U	int	USP code	C	char*	stat code
U	char*	drug_name	U	char*	labeling
C	Status*	status	S	char*	report_name
U	char*	indications	S	char*	observations
U	char*	counter_indications	T	char*	testing_results
			S	int	num lawsuits

In the table, *Name* and *C++ Type* are self-explanatory. *Label* denotes the visibility label of the property, and is the minimum sensitivity level at which the property can be viewed. Drug.status is a direct reference⁷ to an instantiation of Status. Note that the very fact that a reference exists between any Drug and a Status object is itself classified to at least C. This does not interfere with the ability for some Status.labeling values to be accessed at the U level from a Status object, as will be shown.

3.3 Multilevel Object and Identity

TOP objects are generalizations of ONTOS objects, and are likewise differentiated by object identity. Object identity is implemented within the TOP TCB through object identifiers (OID); each client references objects through *tokens* provided uniquely to it by the TOP TCB. TOP *object instantiations* contain multilevel instantiations of some or all of the object's properties. A client that is authorized to 'see' an object instantiation does so through a *view* defined at the client's login level, *l*. This is called the *l*-view of the object instantiation.

⁷In ONTOS DB notation, this would be an OC_Reference.

3.3.1 *l-Instantiation*

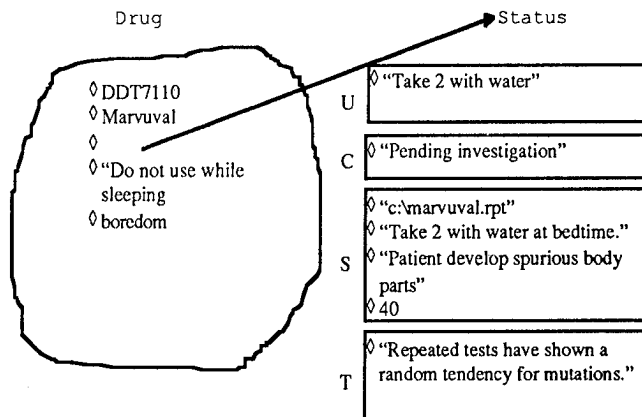
If it exists, the instantiation of an object denoted by the OID and defined at level *l* is called the *l*-instantiation of the object. The *l*-instantiation contains only the data explicitly written at level *l*. In a multilevel ODBMS, object identity remains unique, but the OID may be associated with distinct *l*-instantiations of the object that have been *defined* and entered into the *persistent store* at specific sensitivity (classification) levels. The *complete object* is the full set of *l*-instantiations that share a common OID. The *l*-complete object is the set of *l*'-instantiations of the complete object dominated by *l*.

Associated with each *l*-instantiation is a *semantic vector*. The semantic vector is used by TOP to control and protect the contents of properties in the *l*-instantiation, as well as to provide a means for defining the *l*-view of the object. It also provides the user with the means to enable and maintain cover stories.

3.3.1.1 *l-Instantiation Example*

Consider an instance of Drug with the following values for its U-instantiation: USP_code = DDT7110; drug_name = Marvuval; counter_indications = "Do not use while sleeping". The value of indications was not specified, so the default specified for the property is used. For the C-instantiation we have: indication = boredom; status = 5. The symbol 5 is used to represent a token which itself references an instance of Status. For the S-instantiation we have: indications = acne; counter_indications = "Do not use while thirsty".

For the U-instantiation of Status, labeling = "Take 2 with water." The C-instantiation has stat_code = "Pending investigation." The S-instantiation has labeling = "Take 2 with water at bedtime"; report_name = "c:\marvuval.rpt"; observations = "Patients develop spurious body parts."; num_lawsuits = 40. The T-instantiation has testing_results = "Repeated tests have shown a random tendency for mutations."



In this figure, the amorphous shape on the left represents a C-instantiation of the object Drug. The arrow represents a reference to the *complete* Status object, and shows all of the instantiations in Status. Note that the C-user cannot see the S and T instantiations of this object. The C-user will be presented with a view of these objects based on her client's level, as explained below.

3.3.2 *l-View and Semantic Vector*

Users do not directly access *l*-instantiations. Instead, an *l*-view is dynamically created for the user at the time the object is retrieved. The derivation is based on the elements in the semantic vectors of the *l*-complete object.

Here's how it works: If the *l*-complete object is not empty, then either there exists an *l*-instantiation or there does not.

If there is an *l*-instantiation, then the semantic vector also exists. The *l*-view is built by iterating through the values of the *l*-instantiation's semantic vector. For each property defined at *l*, the value of the property will be determined as follows:

- if the semantic vector element is *static*, the value in the *l*-instantiation is used.
- if the semantic vector element is *scooped*, the value is dynamically determined by the corresponding element in the nearest dominated *l*-instantiation⁸.
- if the semantic vector element is *initialized_scooped*, the value is defined by the schema for the type.

If there is not an *l*-instantiation, then the semantic vector does not exist. The *l*-view is constructed in two steps, as shown below.

3.3.2.1 *MAC simplification (temporary)*

- property values will be scooped directly from the nearest dominated *l*-instantiation.
- property values introduced at *l* and not contained in the nearest dominated *l*-instantiation are treated as though their semantic vector element was *initialized_scooped*, and will acquire default values.

Note that when scooping is used, the property value comes from a dominated *l*-instantiation. TOP uses scooping as a means of ensuring that high-level clients have access to the most current view of data updated at lower levels.

This represents another departure from trusted RDBMSs that support polyinstantiation. In such systems, if a high-level user performs an update, then subsequent updates to lower instantiations of the tuple will be automatically masked by the polyinstantiation. It takes explicit action by the TOP user to enable this form of polyinstantiation, as it can only occur for those properties whose corresponding semantic vector element has been set to static.

Scooping requires the identification of a well-defined source for property values. The complete compartment lattice may potentially contain ambiguities. For example, if a Secret <A, B> property is scooped and distinct values existed at Secret <A> and also at Secret , the scooping would not be well-defined. Several alternatives are being considered for resolution of this problem.

A client operating at level *l* always retrieves an *l*-view (unless specifying otherwise). Any object referenced by this *l*-view is retrieved as the *l*-view of the referent. Since all *l*-instantiations of an object are associated with the same OID, the level of the referent need not be equal to the level at which the reference was originally bound.

3.3.2.2 *Semantic Vector Example*

Continuing from the example above, the semantic vector for the object indicates that nearly all of the specified values are static; all others are scooped. However, since *indication* is specified in the C-instantiation, the semantic vector value is either *initialized_scooped* (if the C-instantiation had been created prior to the U-instantiation) or *static* (if the user intended to create a cover story).

3.3.2.3 *l-View Example*

In the above example the C-user will be presented with a view of *Status* consisting of *labeling* = "Take 2 with water"; and *stat_code* = "Pending investigation." Interestingly, if an T-user were to follow the reference from the T-view of the *Drug* object, he would be presented with the T-view of *Status*, even though the reference was origi-

⁸The TOP TCB ensures that a value always exists when the semantic vector denotes scooping.

nally written at C. This is because all objects are accessed through a token that references the complete object, rather than a specific *l*-instantiation of the object.

3.3.3 Multilevel Inheritance Principle

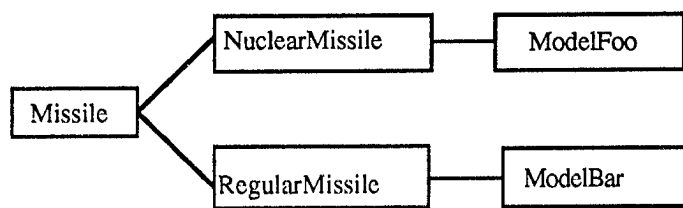
In the schema the explicit definition of an attribute within a type is assigned a visibility label that dominates the level of the type. A property may appear in an *l*-instantiation only if the level of the *l*-instantiation dominates the visibility level of the attribute. This has the following implication on multilevel inheritance: A property in an *l*-instantiation has as its type either

- the type of the object if the attribute is defined explicitly and *l* dominates the visibility level of the attribute, or
- the type of the nearest supertype of the object such that *l* dominates the level of the supertype.

This means that distinct *l*-instantiations within an object instantiation may have different inheritance hierarchies⁹. This causes 'cover stories' for supertype, because the visibility of a supertype in the inheritance hierarchy is constrained by the effective visibility labels of its attributes.

Previous attempts at multilevel object models have been constrained by the requirement that the inheritance hierarchy's sensitivity levels be monotonically non-decreasing from the base type.[1, 6] TOP's approach provides greater flexibility and, therefore, greater richness in semantic expression, with no loss of confidentiality. In particular, this philosophy permits different views of the hierarchy to exist according to a user's clearance. This makes the hierarchy yet another 'property' subject to access control. In effect, the feature introduces schema-level cover stories that can further help to control inferences that may have been based on knowledge of the hierarchy.

For example, consider the following classes:



If the schema is allowed to remain invariant over different visibilities, then we've also exposed the fact that *ModelFoo* and *ModelBar* are distinct in some fashion more generalized than can be justified by the model differences¹⁰. Therefore, it seems reasonable that for those subjects who do not need to know about such distinctions, the schema view should not reveal them. Instead *ModelFoo* and *ModelBar* would appear as direct subclasses of *Missile*.

3.3.4 Procedures

Morgenstern and others have introduced the possibility of having classified procedures ('methods' in the original paper), with the additional potential for several distinctly classified instances of a single procedure to coexist. It is planned that TOP will support the

⁹This concept was hypothesized as necessary in [1], although it ran contrary to earlier models' constraints [5,6].

¹⁰This is assuming that all mnemonic information is removed from the class names, the method names, and the property names. That alone is a significant sacrifice in usability.

specification of multi-instantiated procedures as part of the multilevel schema. Based on the client security level, TOP would bind the proper instantiation to the client domain.

3.4 Updating the Database

Making changes to a database is significantly more complex than viewing its data. The issues of concern are preserving confidentiality and integrity while users are **concurrently** accessing the data through untrusted clients. While these problems are present in trusted relational DBMS, they are more challenging in ODBMS because users are not limited to the use of an interactive query facility such as SQL. Application developers write their own C++ programs to perform customized transactions.

3.4.1 Creation

When a client, logged in at level l , creates a new object instantiation, the TOP TCB creates an OID and generates a semantic vector. The client furnishes the initial values for populating the l -instantiation. If authorized, the client may set elements of the semantic vector. Otherwise, they take on initial values as follows:

- static if l is the visibility level of the property
- initialized_scooped if the visibility level is strictly dominated by l .

3.4.2 Modification

In TOP, updates are subject to DAC, MAC, and integrity constraints that are type-specific and they are further controlled by the semantic vector. There are several cases:

- The client acting at level l , presents modifications to property values that are visible in an l -view of an existing object. If the corresponding l -instantiation does exist and the only modified properties are not scooped (they are, therefore, either static or initialized_scooped), the update is performed to the l -instantiation. This case is completely compliant with the \star -property. All properties to be scooped by this level will continue to be scooped, independent of this update.
- The client acting at level l , presents modifications to property values that are visible in an l -view of an existing object. If the corresponding l -instantiation does *not* exist and the modified properties are not visible in the nearest dominated l -instantiation, the l -instantiation is created and these property values will be placed appropriately according to the settings of the semantic vector, as in the case above.
- The client acting at level l , presents modifications to property values that are visible in an l -view of an existing object. It is possible that all the visibility levels for these property values are lower than l and that they are not yet instantiated at this level. The user may create an l -instantiation dominated by l containing these values. The user can do this with the B3 Trusted Path. A side effect will be that the semantic vector elements corresponding to these properties will be changed from initialized_scooped to scooped.
- In this case, some property value visible below l is updated. This may be because the user intends to create a cover story or it may be because the user wants to modify the property at its level. To create the cover story, the user needs to modify the corresponding element of the semantic vector of the l -instantiation to static. To modify data at a lower level, the B3 Trusted Path must be invoked in order to circumvent the restrictions caused by the \star -property. Both operations may be performed through the Trusted Path: the user may concurrently introduce a lower level cover story and update at multiple levels within the object instantiation.
- A client operating at level l may reference any object for which there exists an l -view. The reference to this object may reside in the l -instantiation of any other object (as defined in the schema).

3.4.2.1 *Cover Story Creation Example*

In order to polyinstantiate a property at level *l*, the corresponding component of the semantic vector element needs to be set to static. The reader may have noted the presence of two cover stories in the example: in the Drug object, *counter_indications* has a cover story at C and a more accurate value at S. In the Status object, *labeling* has a cover story at U and a different value at S. The cover story will be scooped by intermediate levels, so the C-view of the Status object will include the cover story for *labeling*.

3.4.3 *Deletion*

Object *deletion* updates the state of the database. Therefore, like *write*, we cannot allow its observation below the level at which the deleting subject is executing. Furthermore, if there are instantiations above the deletion level, then the deletion is potentially a cover story. Therefore, the effect cannot be automatically cascaded upwards.

The TOP motivation is as follows: to the untrusted user, object deletion, while operating at a particular level, should be indistinguishable from a complete object deletion.

TOP's policy for object deletion is as follows: the level at which the object is deleted is marked by a *tombstone*¹¹. If there are no other instantiations for the object, the complete object is deleted (safely). If other instantiations exist below the deletion level, they continue to remain visible at their respective levels. If other instantiations exist above the deletion level, they also continue to be visible at their respective levels, and any values they scooped from the deleted instantiation would be written upwards to maintain the coherences of such views. Uninstantiated levels of the object appear deleted if their views end up being constructed from a tombstone.

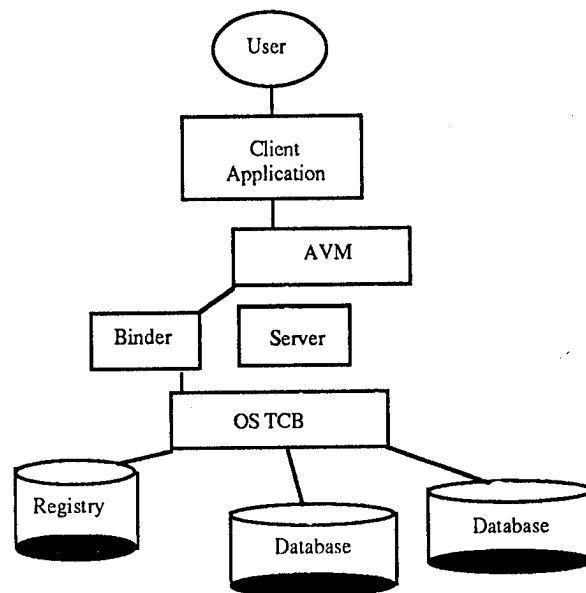
During maintenance, and cover story/polyinstantiation reconciliation, it is possible to "revive" an object (remove the tombstone, usually replacing it with a live *l*-instantiation). Because of the need to maintain the appearance of complete object deletion, we need to insure that any references that used to appear obsolete (pointing to a deleted object) continue to appear obsolete; otherwise, untrusted subjects may infer the existence of higher instantiations. To insure this, we annotate references, and the complete object at each level, with *incarnation numbers*¹². Thus, an obsolete reference continues to appear obsolete, while a fresh references resolves, though both point to the same object.

3.5 **Mediation: the Access Validation Monitor**

The TOP TCB manages all labeling for its objects and is, thus, designed as a trusted subject TCB subset architecture. It is responsible for mediating all accesses between its subjects (the untrusted clients) and its objects. As a client/server architecture, TOP maintains its objects on the fully-trusted server. Mediation is performed through the interposition of the Access Validation Monitor (AVM) between clients and the server. No path is provided between client and server that does not involve the AVM. The AVM and the remainder of the TOP TCB rely on the underlying B1 (or higher) OS/TCB subset to protect their integrity, to authenticate users and their clearances, to identify security attributes of clients and to protect all audit data. It is assumed that the B1 OS/TCB provides a B3-equivalent trusted path mechanism to support private communications between the user and the TOP TCB. The diagram below depicts a logical configuration of the TOP architecture from the user's perspective.

¹¹Tombstones are not visible to the untrusted client, who can only observe obsolete references, and is incapable of distinguishing them from complete object deletions.

¹²Incarnation numbers are not visible to the untrusted client.



One of the critical functions of the AVM is *OID obfuscation*; that is: the minimization of information content of the client's perception of OID, and the minimization of the viability of such information. For every transaction, the AVM produces a new mapping from real OIDs to tokens that are handed out to the client application. The mappings are transient, and vary per transaction. Thus, the information contained in such tokens is relatively short lived (one transaction only), and because of the additional level of indirection, significant OID data patterns are also hidden (sequence and ordering of OIDs, form of the OID, etc.).

4. STATUS AND PLANS

Following an initial investigative study, the TOP project began in earnest in the summer of 1994 and has completed its informal policy model and preliminary prototype design. This policy model and design are undergoing further refinement and implementation of the prototype has begun. An initial configuration is expected for late September 1995. It is planned that a proof-of-concept demonstration on a multilevel database be constructed for delivery to the sponsor in June of 1996. At the time this paper is being written, mechanisms for implementing discretionary access control and audit are being considered and will be reported on in the future.

5. CONCLUSIONS

In this paper, we have discussed the design and design philosophy behind TOP, a research initiative into developing a foundation for a trusted ODBMS. We have explored many of the numerous tradeoffs and considerations needed to support a marriage between confidentiality and integrity, without sacrificing utility. Our differentiation from previous work is manifested by the fact that ONTOS exists today and TOP is being implemented as its next generation.

The authors wish to acknowledge the faith, support, and enthusiasm Joe Giordano has given this project from its naissance. We extend our warmest thanks to ONTOS management, and in particular to Sandra A. Wade for her visionary contributions, unflagging inspiration and support. We would also like to thank Don Marks for his benevolent assistance, and Matt Morgenstern [5], Arnie Rosenthal, and Bill Herndon [1,6] and Win Cuthbert for their insight and frankly-given opinions. Smooches to our seldom-seen spouses and pets.

6. REFERENCES

- [1] Herndon, W., "Can We Do Without Monotonically Non-decreasing Levels in Class Hierarchies?", Unpublished Manuscript, The MITRE Corporation.
- [2] Jajodia, S., and B. Kogan, "Integrating an Object-Oriented Data Model With Multilevel Security", *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, CA, October 1990.
- [3] Lunt, T., "Multilevel Security for Object-Oriented Database Systems", *Proceedings of the 3rd IFIP WG 11.3 Workshop on Database Security*, Monterey, CA, September 1989.
- [4] Millen, J., and T. Lunt, "Security for Object-Oriented Database Systems", *Proceedings of the 1992 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, May 1992.
- [5] Morgenstern M., "A Security Model for Multilevel Objects with Bi-directional Relationships", *Proceedings of the 4th IFIP 11.3 Working Conference in Database Security*, Halifax, England, 1990.
- [6] Rosenthal, A., W. Herndon, B. Thuraisingham, and R. Graubart, "Multilevel Security for Object-Oriented Database Management Systems", Working Paper No. WP-92B0000375, The MITRE Corporation, Bedford, MA, 1993.
- [7] Sandhu, R., R. Thomas, and S. Jajodia, "A Secure Kernelized Architecture for Multilevel Object-Oriented Databases", *Proceedings of the IEEE Computer Security Foundations Workshop IV*, June 1991.
- [8] Schaefer, M., Wade, S.A., *Requirements in Security Policy: Preliminary Informal Access Control Model*, Final Technical Report, National Security Agency, Fort George G. Meade, MD, and Rome Laboratory, Griffiss AFB, NY, 31 March 1994.
- [9] Schaefer, M., Martel, P., Kanawati, A., Lyons, V., *Multilevel Data Model for TOP*, to appear at IFIP, August 1995.

Great Unsolved Problems in Applied Computer Security

What are the great unsolved problems in computer security?

The author proposes four—and announces a \$1,000 prize for the solution to any one of them. The competition, sponsored by FIRST (Forum of Incident and Response Teams), is expected to be an annual event.

In this inaugural year, security experts are challenged to:

- Design a program able to detect the compromise of designated system files, including the program itself
- Develop a fast technique for writing log files to read/write media such that the information, once written, cannot subsequently be modified without detection
- Devise a method to compare the security of two similar computer systems
- Provide a definitive answer to the question, “Who is helped by the full disclosure of details about security holes, and who is hurt?”

The author also explains how the competition will work and how you can submit solutions to qualify for prize money.

Keywords: Security, intrusion, metrics, disclosure.

Mark G. Graff
Sun Microsystems

Voice: 415-688-9151
Fax: 415-329-8258
mark.graff@sun.com

Great Unsolved Problems in Applied Computer Security

What are the great unsolved problems in computer security?

The author proposes four—and announces a \$1,000 prize for the solution to any one of them. The competition, sponsored by FIRST (Forum of Incident and Response Teams), is expected to be an annual event.

In this inaugural year, security experts are challenged to:

- Design a program able to detect the compromise of designated system files, including the program itself
- Develop a fast technique for writing log files to read/write media such that the information, once written, cannot subsequently be modified without detection
- Devise a method to compare the security of two similar computer systems
- Provide a definitive answer to the question, “Who is helped by the full disclosure of details about security holes, and who is hurt?”

The author also explains how the competition will work and how you can submit solutions to qualify for prize money.

Keywords: Security, intrusion, metrics, disclosure.

1.0 How the Competition Will Work

1.1 Challenge

Each year a member of the FIRST Steering Committee will issue the challenge, in the form of a paper presented at the National Information Systems Security Conference. The paper will specify each problem and lay out the allowable parameters of a solution.

1.2 Scope of the Problems

FIRST will select problems from a wide variety of topics such as intrusion detection, network protection, implications of trust among network elements, and sociological elements

of security policy. Each problem will be scaled in such a way that a moderate effort by the right individual or small team may suffice.

No grand theoretical advances in the state of the art are anticipated. No proprietary interest in the solutions themselves may be retained. FIRST seeks only to foster incremental progress along practical lines by the elimination of everyday obstacles to today's practitioners.

1.3 Submissions

Solvers seeking recognition (and money) will submit their proposals as candidate papers for the annual FIRST conference. A panel of experts appointed by the Steering Committee will judge all submissions. Winners will be announced, and prizes awarded, at the FIRST conference.

Each year FIRST will seek an appropriate venue for the publication of selected papers. Submitting authors implicitly agree to such publication and must be prepared to cooperate by meeting deadlines and conforming to editing guidelines. Papers that appear to deal with proprietary topics, or seek to place limits on the distribution of the ideas expressed, will be returned unread.

1.4 Awards

The full \$1,000 prize will be awarded to any solver who either submits a complete solution, or proves that a problem as proposed is intractable.

Occasionally the judges will issue merit awards of lesser amounts to honor those whose contributions, while falling short of the goal, have significantly advanced knowledge in the field. The judges may also decide to split an award among multiple solvers.

Small awards will also be made to those whose suggestions for future topics are accepted.

1.5 Selection

The FIRST Steering Committee will select the problems, based on suggestions from around the world. The goal will be to select challenges which are:

- Practical, difficult to solve, and urgently relevant
- Not already under study, or likely soon to be undertaken
- Of general interest to FIRST members
- Independent of any particular vendor

Problems not solved within a given year may be repeated.

1.6 Common Parameters

Certain parameters will be common to every solution. Software must:

- Run on, or be convertible to, a variety of operating systems, hardware platforms, and file formats
- Require no licensed (third-party or vendor-specific) add-on packages
- Avoid (perhaps as a bonus) the use of any ITAR-restricted (non-exportable) software
- Be robust (i.e., crash-resistant)
- Make practical, appropriate demands on system or network resources

2.0 Problem 1995A: Immaculate Detection

2.1 Problem Statement

Design a program able to detect the compromise of designated system files, including the program itself.

2.2 What's Wrong Now?

No one knows how much is lost today in productivity, time, and trust as a result of computer system intrusions. But few doubt that the toll is significant—and rising.

Programs which detect intrusion by comparing the state of a system to a known-good record are commonplace. Recent improvements in checksum and digital signature schemes have made checking operations more robust. But the state of the art today (1995) requires that the master copy of that system state record be itself protected from compromise. This requirement, often satisfied with the use of a write-protected disk drive, complicates the use of such tools and strongly limits their application.

Is this limitation necessary? Why can't a program be designed which can check itself for tampering? That done, the way would be clear for "unobtrusive" intrusion detection software which could run on a wide variety of hardware and software configurations.

2.3 Parameters

In addition to the common parameters the intrusion detection software must (or must be designed to):

- Operate without the need for write-protected storage
- Operate at user-settable intervals, and be able to run in background or foreground, in steady-state or continual operation

- Allow the system administrator to select the checksumming or digital signature algorithms

For a full award, no manual participation—for example, an operator noticing that an action signifying “all OK” was not taken—can be required.

For extra credit (but no more money) the program should:

- Be adaptive enough not to chatter on voluminously about minor and predictable changes, such as the growth of a log file
- Be alert enough to notice anomalies such as a log file which *shrinks* unexpectedly.

2.4 Approaches

We suggest here two approaches to consider as rough guides (guesses).

2.4.1 Ringing the Changes

Consider the popular Tripwire package (and TAMU's Tiger, similarly). This software meets about all of our specifications except the key one. Would it be feasible to create the checksum database; checksum the database; checksum the checker; then check the checker?

In other words, it may be that the problem can be reduced to a question of whether a particular executable can be produced which can detect its own variance from a predetermined, built-in digital signature.

2.4.2 How Are You? I'm Fine

Another promising approach: using multiple instantiations of the checker software, or cooperating pieces, to check on each other's integrity. The key idea here is that it may not be possible to tamper undetectably and simultaneously with several cooperating processes.

It might even prove workable to operate several such processes around a network, creating a community of software somewhat similar to the Neighborhood Watch program found in many U. S. communities. The TCP-based protocol used on some UNIX systems to synchronize system clocks could be another useful analogue.

Stochastically variable intervals and search extents would seem to add robustness to this model.

3.0 Problem 1995B: Indelible Ink

3.1 Problem Statement

Develop a fast technique for writing log files to read/write media such that the information, once written, cannot be subsequently modified without detection.

3.2 What's Wrong Now?

Today, log files are one of the battlegrounds of system security. Intrusion detection tools often rely on the ability to write audit and log information for later analysis. Intruders, in hiding their tracks, often tamper with log files, to cloak or remove evidence of their activities. Both sides want control of the log files.

Intruders have the edge now, because on most standard systems log files are kept on read/write media. It's hard not to. But how often is it necessary to change (not append to) log files, once they are written?

The challenge is to develop a technique for *log files that last*—that is, the ability to log information on read/write media in such a way that any subsequent modification of the log file is immediately detected.

It's often critical to know *when* events happened, too, and in what sequence. Protection of the system clock is beyond the scope of this problem, but it's imperative to protect the sequence of the records (and any time stamp information) as carefully as the rest of the logged data.

3.3 Parameters

In addition to the common parameters the logging software must (or be designed to):

- Be efficient (not impose a severe performance cost)
- Allow the system administrator to select the verification algorithms
- Be verifiable for correctness of operation

A bonus would be the ability to detect interruption of the process that is writing the log file.

As another desirable feature the program could offer the ability to repair log files which have been tampered with. This would seem to require either redundant record-keeping, or special buffering by the process responsible for logging.

3.4 Approaches

Target system utilities such as UNIX's `syslog` or VMS's `OPCOM`. The external interfaces could remain the same, and the internal operations be modified to perform integrity checks prior to appending each log entry.

Simple enough. But how can you protect against changes?

- Look into a "check digit" approach, associating a checksum with each log entries.
- It might be possible to come up with a scheme of redundant loggers or log entries such that an attempt to introduce a variation between the two would raise an alarm.
- Information kept in two differing states should be harder to change undetectably at the same time. (We don't mean one copy in Maryland and the other in New York. Try, for example, one in process memory and one force-written to disk.)
- There's no reason redundant records must have the same format. Maybe keeping information in different formats (say, one copy in plain text and one encrypted) would make it harder to sneak through a change.

4.0 Problem 1995C: In Numbers There Is Safety

4.1 Problem Statement

Devise a method to compare the security of two similar computer systems.

4.2 What's Wrong Now?

To paraphrase Ernest Thompson, the master engineer who laid the first transatlantic cable, "To measure is to know." When it comes to the relative security of two machines, we know nothing.

Trying to improve the security of a system without being able to measure the result of your changes is like pushing a rope. There's effort; there's apparent progress; but someday you're likely to trip over the result.

We need to be able to compare the relative security of:

- Two similar systems, even if they come from different manufacturers
- The same system after a patch or other putative fix has been applied
- A system before and after a major software upgrade or configuration change has been applied

Notice, too, that once there's a way to determine which of two systems is more secure, the road is clear to useful spin-offs such as benchmarks and other metrics. These are going to be terrific; but the comparison operator has to come first.

To keep it simple let's restrict the arena to software. Hardware, today, is not the problem.

4.3 Parameters

In addition to the common parameters the method must:

- Produce unambiguous and reproducible results
- Take into account the many different security environments, e.g., behind a firewall
- Not become outdated with the discovery of every new vulnerability
- Not itself represent a security risk (as a straightforward system canvas or vulnerability inventory might)
- Allow the system administrator to select from among many comparison criteria

For partial credit the comparison could be restricted to a subset of threats.

4.4 Approaches

The parameters seem to imply the use of some outside agent. Perhaps the winning technique will:

- Be an extension of today's penetration studies and tiger teams
- Allow (and require) the creation of a secure, contractible network audit service
- Require the development of a technique for opaquely testing a system for a vulnerability, using a program of assured integrity that cannot meaningfully be monitored during operation, nor reverse engineered when static

5.0 Problem 1995D: Do What I Say and Nobody Will Get Hurt

5.1 Problem Statement

Provide a definitive answer to the question, "Who is helped by the full disclosure of details about security holes, and who is hurt?"

5.2 What's Wrong Now?

The debate over "full disclosure" has of late filled all of the security newsgroups and mailing lists with opinions. Is "security by obscurity" dead? Does information want to be free?

Some system administrators argue that the disclosure of detailed information about security holes puts ammunition into the guns of those who would break into systems—and that releasing an exploitation script supplies a complete, loaded weapon.

Others counter that the advisory bulletins issued by FIRST response teams are useless, because not enough information is supplied either to reproduce the problem or design and test a fix. It's important to balance short-term versus long-term interests, too. Full disclosure of exploitation details may cause immediate disruption; but it also tends to ensure that the bugs so revealed, once killed, will stay dead. Some examples that buttress this reasoning are the "sendmail wizard" and "finger overflow" problems.

While the debate rages on, many systems are being broken into with the use of "fully disclosed" vulnerabilities, while perhaps as many are being trashed or tampered with because the people charged with their protection are themselves being "protected" from knowing how to fix them.

Let's try to get this settled. Everybody in the field has an opinion. What are the facts?

5.3 Parameters

Answering the question may sound easy. But we need *numbers* at least as much as we need analysis.

For a full award, the following information must also be supplied, with justification:

- Close estimates of the vendor, vendor-dependent, and roll-your-own populations
- A characterization of the populations (sophistication, software they're running, platform, etc.)
- Weighted risk analysis, broken down by population (e.g., risks for vendors)
- An analysis of which times at which disclosures are most damaging
- An analysis of possible warning periods. Is two weeks better than two months?
- Proposals for a *modus vivendi*, a practical set of arrangements allowing these groups with differing information needs to cooperate (or at least co-exist)

5.4 Approaches

We anticipate that one way solvers might try to get real numbers is to conduct a real-world experiment, using standard double-blind protocols. Such investigators would assume sole risk and responsibility if the experiment misfires. FIRST is specifically not encouraging an industry-wide hoax.

6.0 How to Submit an Entry

FIRST maintains the *unsolved-problems@first.org* mail alias for interested parties. Use this to get a copy of the contest rules, get more information about the problems, and exchange ideas with other participants.

With a solution or partial solution in hand, submit your work as a paper to the upcoming FIRST conference. The 1996 conference will take place in the San Francisco Bay area.

For information about how to prepare and submit a paper to the FIRST, watch for the annual Call For Papers.

7.0 Acknowledgments

The author wishes to thank the members of the 1994-1995 Steering Committee for their confidence and assistance, and also:

- Problem submitters Gene Spafford (of Purdue) and Danny Smith (AusCERT)
- Advisers Steve Weeber (CIAC) and Roman Galperin (Sun Microsystems)

ADDRESSING INFOSEC ANALYSIS PROBLEMS USING RULE-BASED TECHNOLOGY

Richard B. Neely, Ph.D.
rneely@cos.cta.com

James W. Freeman, Ph.D.
jfreeman@cos.cta.com

CTA INCORPORATED
7150 Campus Dr.
Colorado Springs CO 80920

Abstract

Obtaining the necessary understanding of the security properties and vulnerabilities of systems, which are becoming ever more complex, requires significant analytic effort. A security analysis team (either development or evaluation) needs to navigate through large amounts of documentation, partition significant problem domains, and simulate or emulate a system or particular components, to make engineering based statements about a system's security.

Current approaches, techniques and supporting tools, including third-generation based technologies are helpful, but are not sufficient. Method specific techniques based on CASE environments are often too narrow for addressing the wide variety of security issues faced by an analyst.

This paper discusses an approach to some different types of security problems and our experience in using a rule-based technology that is not method specific and that has contributed to improved understanding of the specific problems. The use of the technology appears to enable an analyst to address a wide-variety of issues in the problem domains of the analyst without forcing the analyst to become an expert in the underlying rule-based technology.

1. Introduction

Systems are becoming increasingly interconnected with more functionality. The need for a security analyst to analyze, understand, and explain a system's security

mechanisms and vulnerabilities is increasingly challenged by a system's complexity. The complexities and interrelationships of present systems can easily overwhelm a security analyst using current approaches and supporting technology.

For example, within most current approaches, an analysis team makes security statements about a system's underlying security architecture. That structure is often based on a monolithic trusted computing base (TCB). The analysis team then implicitly extrapolates, with some level of assurance, the structure to a collection of security statements about the full system. This has been a credible approach when the TCB was relatively simple, monolithic, and the step to the full system was not large.

Increasing functionality and connectivity adds complexity to networked and distributed systems. The resulting underlying security architecture, including a TCB, also becomes complex and distributed. Also, the step to the full system from the TCB often is either larger than anticipated, in order to keep the TCB manageable for understandability (particularly for embedded systems), or else the resulting TCB itself is significantly larger and more complicated than desired. Such complex systems are necessary in order for systems with critical security requirements to exploit advances in other system and software engineering disciplines [1].

Each component within a distributed system may have undergone a thorough security analysis relative to the system

architecture. Nevertheless, residual vulnerabilities in the high reliability components, combined with potential exploitation by less reliable components, could result in security failures. These might include data compromise via system output interfaces or loss of data or system integrity. In other words, the system structure itself must be considered in a security analysis [2]. Using currently available methods, a security analysis team would probably be unable to say, with a reasonable level of assurance, something definitive concerning the security of the set of end-to-end data flows for such a system. That would be true, even though they used the best available technology, including current methods and supported by a CASE tool environment.

As has been described by Hirsch [3], a security analyst needs to *navigate* through large amounts of documentation with more than an automatic page-turning capability. An analyst needs to *partition* a general problem into smaller problems and maintain or preserve relationships among the parts. An analyst also needs to apply appropriate *metrics* to system/software entities. An analyst also needs to *reverse engineer* available information to either generate missing pieces or to double check whether an entity is appropriately derived from higher-level specifications. An analyst performs a *verification* that establishes a correspondence between higher-level and lower-level specifications. Finally, an analyst, by developing a *simulation or emulation*, can obtain value by observing the behavior of a system under controlled conditions.

What is needed, therefore, is a technology or a well-integrated collection of supporting technologies that enables an analyst to accomplish the identified activities in an efficient and effective manner. This paper identifies an approach, our recent experience, and near-term plans that address aspects of a security analyst's navigating, partitioning, reverse engineering, verifying, and simulating a

system to identify and demonstrate vulnerabilities. The approach to INFOSEC analysis problems uses rule-based techniques.

One problem that we have addressed in this way is providing assurance for system security related to end-to-end system flows—the example mentioned above. This problem has an important application within the security analysis of MLS tactical systems.

A second problem we have addressed is how to control disclosure of inferred information from a relational data base system, while still enabling the system's users to accomplish their mission. An effective solution can be expected to be quite complex, involving examination of multiple criteria in parallel and using expert system techniques.

It is important to observe that the second problem is an INFOSEC problem that is essentially unrelated to sensitivity labels. One consequence of these two problems is that a very flexible environment would be needed by an analyst to solve both these problems effectively, using the same environment for both.

Following this introduction and problem identification, this paper discusses the use of rule-based environments (Section 2). Following that discussion is a description of our experiences and results in developing rule-based techniques and tools to solve INFOSEC problems (Section 3). Finally, we present the conclusions that we have drawn on the basis of those efforts (Section 4).

2. Problems and Solution Directions

2.1 Issues In Automated Analysis for INFOSEC

INFOSEC analysis has mostly used manual methods, with any appropriate software development tools. Manual methods remain common, though some limited support tools have been developed. Manual analysis has been acceptable in the past, but newer systems are large and complex. For such systems,

manual methods are too expensive, unreliable, and unrepeatable. Repeatability is at issue because of multiple builds and development blocks, for which analysis must be repeated. As a result, some significant risk is often accepted in systems. Further, the concern for many systems is not that there is a high security risk, but that, with current practice and technology, the risk is unknown and not definable in a measurable way. This is because of the intractability of effective manual analysis.

To have an effective as well as cost-effective automated analysis, a solution ought to have the benefits of manual analysis (e.g., flexibility and an expert's base of knowledge) without its shortcomings. Past approaches to automating the analysis, while in some ways are beneficial, are also inadequate. For example, simple object-oriented approaches, while providing some useful data modeling, do not address process modeling. Consequently, they do not go nearly far enough in providing proximity between problem and solution spaces.

One way to look at this is that one wants to see the "big picture"—what's "really going on," providing a framework for important details, rather than seeing details in a way that only obscures. That is the situation in the game of chess. A beginner, or even intermediate, player sees 64 squares on a playing field and 16 chessmen; a chess master sees a "position" made up, perhaps, of three or four high-level units. The master's resulting analysis of the game provides a deeper understanding than the details can provide to the novice.

This illustration applies specifically to INFOSEC: a simplistically algorithmic approach can inhibit the solution of some problems, because the problems are obscured by complexity that is not fundamental to the problems. A rule-based approach can allow an analyst to logically integrate all aspects of a system without the analyst's drowning in detail.

The big-picture view provides improved understanding of a complex system, independent of whether rule-based automation (or any automation) is considered. For example, Bodeau's system-of-system perspective for risk analysis would illuminate the risk analysis problem addressed, even in the absence of the ANSSR tool [4].

Past tools have used the extensive run-time libraries associated with their implementation languages, but these libraries do not provide an adequate environment to facilitate reusability. In developing INFOSEC tools in the past, much design and implementation energy has often been focused on the infrastructure, at the expense of solving the problem. For example, software tool development may consume as much as 90% of the available resources in providing support for a graphical user interface (GUI) [5]. What is required is an environment for automating INFOSEC analysis with an infrastructure built in, that developed tools can inherit.

2.2 A Solution: Rule-Based Automation

An examination of rule-based support for INFOSEC analysis appears to indicate that this is an approach that could solve some of the analysis problems. Some rule-based environments do exist that have been shown to be applicable.

2.2.1 Rule-Based Concepts

A rule-based environment provides flexibility of data modeling beyond what is provided by an object-oriented language. That is because the underlying capability for data description is more expressive (e.g., set theory instead of standard programming language types with narrow extensions).

Process modeling is also possible in a rule-based environment. Process modeling allows specification of what is to be done (versus how to do it): it allows expression in terms of rules, which preserves the best aspects of manual processing. An important aspect to preserve is

adaptability as analysis needs change. In particular, process modeling is critical for proximity of problem and solution spaces.

Further, a general rule-based environment supports the separation of process and data, which is critical for robustness, flexibility, and extensibility.

2.2.2 Benefits of a Rule-Based Approach for INFOSEC

An important area best supported by rule-based approaches is logical integration, which involves both information integration and development tool integration. A rule-based environment can provide a capability for both aspects of logical integration. Data models can be merged and rule-based methods devised to relate differing sources of assurance [6]. These include risk analysis, hardware security characteristics, physical security, and communications security (COMSEC) characteristics.

A general rule-based environment is also ideal for integrating a variety of tool-based techniques for security analysis. Such integration involves developing common data models of information processed by each of the techniques, and providing data capture release interfaces with each. This allows many existing capabilities to be used, rather than reinventing them. Examples are CASE tools, configuration management tools, configuration management tools, and specialized security analysis tools, such as Romulus (discussed below).

2.2.3 Lack of Generality in Some Rule-Based Environments

To be useful for INFOSEC applications, a rule-based environment must not limit the data model or method. These must be open-ended, so that the developer is able to specify them to address the specific application problem. This is not a specious point, for most rule-based environments are designed with a built-in method. Even an environment oriented toward INFOSEC that has a built-in, non-extensible

method has limited value. As an example, computer-aided software/system engineering (CASE) environments are specific to software or system engineering and not related to INFOSEC analysis. Note that CASE tools are not rule-based *environments*, but rather specific rule-based *tools*.

An example of a rule-based tool designed for security is Romulus, developed by ORA. Romulus addresses a range of security problems, and is effective within that range. Even so, Romulus supports a built-in method that is definitely narrower than the full range of security problems. For example, it would be very difficult (and perhaps ineffective) to apply Romulus to the problems described in Section 3 (particularly the disclosure control problem), or to any other problem that is unrelated to security labels. This is because the method implemented by Romulus is label based. The method of Romulus is based on the concept of "restrictiveness," and specifically the "Hookup Theorem" [7]. The built-in theorem prover of Romulus extends its usefulness, but that does not have the flexibility of interfacing with an arbitrary external theorem prover, such as Computational Logic, Inc.'s, Nqthm (Boyer-Moore Theorem Prover) [8]. Because of the built-in method nature of Romulus, that would not be possible without significant redesign.

A particular limitation of Romulus is that a security model (or at least a limited family of models) is built in, and therefore an inferred policy is assumed. This policy is based on label-based access, restrictiveness (upper bound on the sensitivity of system output data), and hookup (composition) of system components. Even so, the Romulus view is more general than many security approaches, since it is able to deal with non-deterministic systems, such as distributed system with concurrent processing.

Despite the advantageous features of the Romulus approach, with its techniques and tool support, Romulus has a serious limitation. The

limitation is that it is method specific and is not easily extensible to other methods. Further, the tools do not possess lower-level support interfaces, in an open-system architecture, that would allow developers to produce and integrate tools for other methods, without pervasive redesign.

2.2.4 Applicable Rule-Based Environments

A general rule-based environment should separate information representation and processing from user interface support. For a GUI, many high-level primitives should be available without unduly constraining the final nature of the interface. Thus, what is needed is an environment that has a variety of processing and user interfacing building blocks without pre-defining actual methods or interfaces. Because of the variety of security concepts that it might be desirable to support, the environment itself should *not* be sufficiently security oriented that it prescribes aspects of any particular security method.

Several rule-based environments exist that, according to marketing information, appear to have capabilities that meet the above criteria. They are [9]:

- Level5 Object, from Information Builders, Inc.;
- Nexpert Object, from Neuron Data;
- Kappa, from Intellicorp;
- ART*IM from Inference; and
- Virtual Software Factory (VSF), from Integrated Software Development Environment (ISDE) Metaware, Ltd. (in the UK) [10].

According to the information available to us, these environments have similar capabilities; we are in the process of obtaining additional information about them. Our experience is entirely with VSF, and has demonstrated that VSF has the required features to support INFOSEC analysis (and a variety of other applications). The technology focus of this paper is demonstrated by our VSF experience.

The basis of the data modeling notation within VSF is set theory, from which a wide variety of object types can be defined. VSF provides for several kinds of set definitions. Rules defining a method are expressed as set membership constraints and set membership consequences. The constraints determine what members may be asserted into or deleted from particular sets in particular contexts. The consequences cause automatic assertions and deletions based on preliminary assertions and deletions.

VSF provides capabilities for data capture and for reporting that are independent from data representations. A single, pervasive knowledge base is maintained, and so every data view into the knowledge base is always consistent with every other. These capabilities exist in both graphical and textual forms, both of which may be used interactively and non-interactively. Reports are generally non-interactive only.

This section began with a description of problems associated with manual INFOSEC analysis methods, and with criteria for automating INFOSEC analysis. Our experience has shown us that a rule-based analysis approach (particularly using VSF) satisfies those criteria. Section 3 presents some details of our experience in this area, and substantiates the claim that our approach indeed satisfies the criteria for automating INFOSEC analysis.

3. Experience and Results In Solving INFOSEC Problems With VSF

This section reports some of our actual experience using VSF to address a variety of INFOSEC problems. The benefit of reporting this experience is that it provides an awareness of some of the kinds of INFOSEC problems that can be solved using a general rule-based environment, and may encourage others to try such an approach. In addition to reporting success in certain areas, we feel that evidence

is provided that some INFOSEC problems can be solved better with a rule-based approach than with a more traditional approach. By "better" we mean a solution that is complete, more extensible, more understandable, and possibly more efficient.

3.1 Data Flow Security Analysis

Within CTA's system security engineering work on a major tactical MLS system, we have encountered a concern regarding certification of the system voiced by the security evaluation team. The concern is that in order to avoid the necessity of making the size of the software portion of the trusted computing base (TCB) extremely large (i.e., a sizable fraction of the entire software of the system), an additional source of system security assurance would be needed. The goal of the assurance is to reduce the risk of data compromise due to data flow out of the MLS system.

The specific problem is that within the system, a large proportion of components handle data at multiple security levels and are capable of writing high data to low destinations (e.g., files), given their context within the system architecture. Without further analysis, all such components must be trusted not to do so, and therefore are within the TCB. That would be an untenable situation. It would mean that the majority of the software within the MLS system would be in the TCB, and there are not sufficient resources available to provide the necessary security analysis.

The additional source of assurance proposed is to examine each end-to-end path to determine, given the sensitivity of data entering the path and allowed to leave the path, whether a combination of components on the path creates a risk of data compromise. The idea is that a large number of multilevel components might in that way be determined to be in no position to compromise data with respect to any end-to-end path, and so not to be within the TCB. A problem with developing a tool for such end-to-end analysis is that the MLS

system is very complex, with thousands of components, and perhaps tens or hundreds of thousands of distinct end-to-end paths. It seemed that any third-generation solution would be intractable, or at least would require significant research effort for development.

We felt that certain of the capabilities of the VSF environment could result in a solution that would require tractable effort, would provide reasonable performance, and would help produce a more effective security analysis than by manual means. Accordingly, we developed a proof-of-concept tool to solve a simplified version of the problem. Our expectation that VSF would provide an appropriate solution platform was based on several observations. First, it seemed that VSF's ability to deal with data models and to express rules that would provide for apparently parallel analysis would allow for a clear solution. Second, based on our previous experience with software re-engineering using VSF, we knew that the advertised high performance of VSF's knowledge base was a reality. Finally, certain aspects of a solution that would be unacceptably complex to express algorithmically already existed within VSF as set membership manipulations of various kinds.

An example of the latter is VSF's transitive closure set definition to deal with data flow paths. It is necessary to understand, in order to follow this example, that the data modeling paradigm of VSF uses standard mathematical set theory. In order to define and manipulate data flow paths, we treated each component of the system as a node in a graph, and direct data flow between a pair of components as an arc in the graph. Then, within VSF, we represented each component node as a member of a primitive (unstructured) set, *Component*, each data flow arc as a member of the Cartesian product set $Flow = Component \times Component$. We were then able to define the set $Path = closure(Flow)$. Once the *Component* and *Flow* sets were populated (thus defining the

system architecture), the set *Path* would automatically represent all data flow paths (end-to-end and otherwise).

It would be incredibly inefficient to fully populate *Path*, to represent all the data flow paths in the system, and indeed VSF does not do so. It merely determines what particular elements are in that set based on particular knowledge base queries. This was significant when a particular rule was developed to determine the set of partial paths on which lay both a multilevel component and a potential upstream exploiter (i.e., a uni-level, untrusted component processing classified data). In that circumstance, exactly the appropriate members of *Path* were extracted to populate that partial path set. No other members of *Path* were ever created.

The proof-of-concept tool was successful. We were able to demonstrate the tool not only to audiences related to the tactical MLS system, but also to other groups. Because of the clarity of the tool itself, people with no familiarity with the MLS system were able to understand the tool implementation easily. That response was expressed in terms of its GUI, its usability, and the ease of understanding of its implementation in VSF

It is noteworthy that as we began to study the security flow analysis problem, existing statements of the problem were incomplete and often confused. Indeed, it was not until we began to formulate VSF rules to specify a solution to the problem that we began to understand the problem ourselves. This was not unexpected, in that often a rigorous analysis helps clarify a problem.

We expect that a full version of the tool, capable of managing all aspects of a security data flow analysis for the MLS system, would be a success, based on the proof-of-concept prototype. This is based on the prototype's satisfying (within its limited scope) the completeness, extensibility, understandability, and efficiency properties (mentioned above).

While the prototype is not intended to be complete, its ease of extensibility (based on the addition of certain capabilities in the knowledge base and in the user interface) should assure ultimate completeness. Responses to demonstrations and explanations of the tool indicate that the understandability property is satisfied. Full-size (or even nearly full size) knowledge-base populations have not been applied to the prototype, but our previous experience with re-engineering using VSF indicates the efficiency of a full version of the tool applied to the entire MLS system.

In terms of the needs of a security analyst described by Hirsch (as indicated in Section 1), the developed tool primarily addresses the verification and simulation needs. The tool, in effect, simulates the flow of data through the end-to-end paths across the system. It also verifies whether pre-defined TCB membership characteristics are consistent with other system characteristics, including system architecture. It is anticipated that a production version of the tool, if developed, would support partitioning the data flow problem to allow multiple analysts to work on the problem concurrently.

3.2 Data Base Disclosure Control

There is a significant issue in developing concepts and techniques to deal with controlling disclosure of sensitive information in relational data bases. Certain kinds of sensitive information, when released in "small doses," may not result in a security risk. At the same time, larger amounts of the same information may result in quite significant security risks. This concept of risk based on amount of data released is known as the aggregation problem. When information is released only indirectly via logical analysis of the information permitted to be released, this is known as an inference attack. Information released via an inference attack is also an example of aggregation if it is more sensitive than any (quantity of) information that would

have been permitted directly. These concerns are explained in detail in [11].

The approach to disclosure control that we addressed is intended to shut down such aggregation and inference channels in a way that does not place unnecessary restrictions on access to the data base. The approach, developed by Motro, Marks, and Jajodia [12], is to limit accesses only based on predefined data base views, termed *concepts*. A numeric threshold is defined for each concept, and data base tuples that relate to each concept are allowed to be released only up to the number corresponding to the threshold. Any number of such concepts may be defined. A "lifetime" count is maintained for each concept, for each user. This approach is far more precise than the relatively naive concept of counting tuples across a whole data base table, or by comparing exact queries. The precision means that the exact information to be protected is indeed protected, but no more.

This disclosure control approach appeared to us to be ideal for implementation using VSF. One important characteristic of the approach is that it is fundamentally parallel in concept. Accordingly, the view taken in a VSF implementation, which involves performing an operation on all the elements of a defined set (without regard to sequentiality or order), would relate well to the problem space. Further, because of VSF's capability for supporting interfaces to arbitrary external systems and forms of data, a VSF-based disclosure control tool could be made to interface with a relational data base management system (RDBMS) without undue effort.

For the prototype disclosure control tool that we have implemented, we simulated a RDBMS within VSF itself, rather than interfacing to an external RDBMS. This was because we did not have available a RDBMS that would run on the platform (Intel 80486 running OS/2) on which we were developing

the tool. It was, in any case, instructive to see the limited amount of code (rules) required for that simulation—about a page. Given that the authors are not very experienced with development in VSF, that is probably not at all the most efficient or compact possible simulation of an RDBMS in VSF!

The resulting tool is generally a success with respect to the same properties as considered for the data flow analysis tool: completeness, extensibility, understandability, and efficiency. The tool performs precisely the disclosure control function originally specified. Extensions to that functionality have been identified, and based on preliminary analysis, we are convinced that corresponding extensions to the tool will be easy to make. This is primarily because of the automatically abstract nature of data and method descriptions in VSF. Demonstrations of the tool, including to the customer, were successful. In particular, the functionality as made visible by the easily developed user interface that described the results of the RDBMS queries was understandable and fully acceptable. Further, the behavior of the disclosure control functionality was accepted as faithfully implementing the disclosure control method.

Those to whom demonstrations and explanations of the implementation of the tool have been given—including the customer, who has no background with VSF—have expressed a reaction that the implementation itself is easy to understand, in terms of the data modeling and the rule-based method definition.

This tool also addresses Hirsch's verification and simulation needs. The direct need addressed is verification. What is being verified is the disclosure control method itself by implementing it and viewing the results. In order to test the method effectively, it was also necessary, in the circumstances, to simulate a RDBMS. In retrospect, using a simulated RDBMS was beneficial to verifying the method

because it allowed a more controlled execution environment.

Our description of the disclosure control method in terms of VSF rules clarified the method itself, and, further, resulted in the discovery of some ambiguities in the method as originally described. It is likely that, after some debugging, the ambiguities would have been discovered as a result of a third-generation implementation in a language like C or Ada. In contrast, the ambiguities were immediately manifest upon expressing the method in the form of VSF rules.

3.3 Future Applications

We are at present planning two additional INFOSEC applications of VSF. They are:

- implementation of a representation of Hoare's process external traces that we have used in past security analysis efforts [13]; and
- security evaluation support, involving the integration of specialized software development and security analysis tools (such as the Boyer-Moore theorem prover [8]).

We now discuss briefly how we anticipate dealing with the first of these using the VSF environment.

In [13], we have described a security analysis method, which we term Boundary Flow Analysis (BFA), which we have used successfully for several security analysis projects. The method is related to Hoare's process external traces, and has been provided with a useful notation by Moore at the Naval Research Laboratory (NRL) [14]. Also related to BFA is NRL's "assumptions and assertions" security certification approach [15].

The concept of BFA is to view a system in terms of multiple levels of refinement and in terms of a data flow diagram at the same time. At each represented level of refinement, a logical history of information (treated as a sequence of information units) entering and leaving each component at each interface is

maintained. Security requirements are expressed for each component (including the system itself) in terms of the interface histories. Verification methods are applied to show that if all the security requirements of lower-level components are satisfied, then the security requirements of upper-level components are satisfied.

Having had success with this approach using the Gypsy Verification Environment, we decided it would be valuable to implement the approach within VSF. To date we have implemented a portion of the BFA approach, but do not have a fully operational tool.

4. Conclusions

This paper has outlined our experience in applying an approach to INFOSEC analysis problems using a rule-based technology and environment, specifically that provided by the Virtual Software Factory. We note that developing a tool for each of the problems discussed took approximately 3-4 man-weeks of effort, which included the developer's learning aspects about the underlying VSF environment. It is important to note that the tool developer is not certainly a VSF expert. Considering the results stemming from a variety of problems faced by an analyst, developing tools using VSF, and applying the tools to solve the problems, we feel that such an approach can be effective. Customer feedback via demonstrations has validated this conclusion. Finally, we have identified some of the directions planned in applying this promising technology.

5. References

- [1] J. N. Froscher, M. Kang, J. McDermott, O. Costich, and C. E. Landwehr, "A Practical Approach to High Assurance Multilevel Secure Computing Service," *Proceedings of the Tenth Computer Security Applications Conference*, Orlando, Florida, December 1994, pp. 2-11.
- [2] R. B. Neely and J. W. Freeman, "Structuring Systems for Formal Verification," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, April 1985, pp. 2-13.
- [3] S. J. Hirsch, "Software Analysis Requirements: An Abstract View," INFOSEC Standards and Evaluations Group, C62, Office of INFOSEC Evaluation Technologies, Computer Science Division, National Security Agency, June 1992.
- [4] D. J. Bodeau and F. N. Chase, "Modeling Constructs for Describing a Complex System-of-Systems," *Proceedings of the Ninth Computer Security Applications Conference*, Orlando, Florida, December 1993, pp. 140-148.
- [5] ISDE Metaware, Inc., "Virtual Software Factory Overview," 1992.
- [6] R. B. Neely and J. W. Freeman, "Rigorous Integration of Sources of Assurance," *Proceedings of the Conference on Computer Assurance (COMPASS)*, Washington, D. C., July 1986, pp. 100-110.
- [7] I. Sutherland, T. Korelsky, D. McCulloch, D. Rosenthal, J. Seldin, M. Lam, C. Eichenlaub, B. Esrig, J. Hook, C. Klapper, G. Pottinger, O. Rambow, and S. Perlo, *Romulus: A Computer Security Properties Modeling Environment (Overview)*, RL-TR-91-36, Vol. 1, ORA for Rome Laboratory, April 1991.
- [8] R. S. Boyer and J. S. Moore, *A Computational Logic*, New York, Academic Press, 1979.
- [9] Datapro Information Services Group, "Information Builders, Inc., LEVEL5 OBJECT," *Datapro Computer Systems Analyst*, McGraw-Hill, Delran, New Jersey, 1995.
- [10] ISDE Metaware, Inc., *User Documentation for the VSF Methods Workbench*, VSF-MWB Version 3.9, 1994.
- [11] T. F. Lunt, "Aggregation and Inference: Facts and Fallacies," *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, May 1989, pp. 102-109.
- [12] A. Motro, D. Marks, and S. Jajodia, "Aggregation in Relational Databases: Controlled Disclosure of Sensitive Information," *Proceedings of ESORICS 94*, 1994.
- [13] J. W. Freeman, R. B. Neely, and M. A. Heckard, "A Validated Security Policy Modeling Approach," *Proceedings of the Tenth Computer Security Applications Conference*, Orlando, Florida, December 1994, pp. 189-200.
- [14] A. Moore, "Specification and Verified Decomposition of System Requirements Using CSP," *IEEE Transactions on Software Engineering*, Vol. 16, No. 9, September 1990, pp. 932-948.
- [15] C. N. Payne, J. N. Froscher, and C. E. Landwehr, "Toward a Comprehensive INFOSEC Certification Methodology," *Proceedings of the 16th National Computer Security Conference*, Baltimore, Maryland, September 1993, pp. 165-172.

IDENTIFICATION OF SUBJECTS AND OBJECTS IN A TRUSTED EXTENSIBLE CLIENT SERVER ARCHITECTURE

Terry C. Vickers Benzel
Trusted Information Systems, Inc.
11340 W. Olympic Blvd., Ste 265
Los Angeles, CA 90064
310/477-5828
tcvb@la.tis.com

E. John Sebes
Trusted Information Systems, Inc.
444 Castro Street, Ste 800
Mountain View, CA 94041
415/962-8885
ejs@ba.tis.com

Homayoon Tajalli
Trusted Information Systems, Inc.
3060 Washington Road
Glenwood, Maryland 21738
301/854-6889
tj@tis.com

Abstract

Trusted Mach (TMach¹) is a trusted operating system with a type extensible framework supporting a client/server architecture. The TCB implements the type framework and provides trusted system services within it. The framework is extensible: untrusted client software can define and implement new types using the same underlying microkernel mechanisms that the TCB uses to implement its types. To client software there is no visible difference between objects implemented by the TCB and objects of untrusted application servers. From a TCB modeling point of view, however, the difference between these two kinds of objects is critical. The definition of the subjects and security-objects of the system extends the TCSEC paradigm to encompass the system's extensibility. The paper presents an overview of TMach, a definition of its subjects and security-objects and an account of the assurance of the system as related to the type-based client/server architecture.

Keywords: Extensible, client server Trust, distributed systems, Mach, B3.

1 Introduction

Developers of the TCSEC recognized the importance of clearly identifying the set of subjects and objects to be controlled by the TCB. This fundamental notion was derived from process-based architectures of the trusted operating systems of the day, which were expected to consist of a monolithic security kernel and a collection of trusted subjects. Subjects were closely tied to executing processes, and objects were containers of information managed by the security kernel.

This foundational view of trusted systems is being updated by new and emerging client/server architectures of microkernel-based systems. The microkernel basis allows multiple independent servers to implement system services, while allowing the microkernel to implement only the most basic system mechanisms. The microkernel/server architecture is inherently extensible, so that new servers can be added to implement new services. Furthermore, these new services may be either system-level services or application-level services.

Modeling subjects and security-objects in the context of type extensible client/server architectures is a new and critical aspect of modern trusted system development. The model must describe the security features of the extensibility mechanisms. This report describes one approach to extending the TCSEC modeling concepts to a trusted client/server system with secure extensibility that derives from the microkernel basis.

1.1 Basic Subject/Object Definition Then ...

In early trusted systems such as the Honeywell SCOMP and Multics systems, the TCB consisted of a trusted kernel and a small collection of trusted processes. The security kernel created and managed all subjects,

¹Trusted Mach and TMach are Registered Trademarks of Trusted Information Systems, Inc. (TIS)

which were simply processes executing on behalf of logged-on users. The security kernel also created and managed all objects and enforced controls of access of them by subjects. Object were typically passive containers of information, like memory segments and devices.

Subjects accessed objects only via the kernel, and the various kernel interfaces all rested on one mechanism—a call instruction—which trapped to a kernel gate. Only by this mechanism could a process request that the kernel perform for it some access to some object. Trusted processes used this same mechanism, the only difference being that a trusted process might have some privilege which would cause the kernel to treat the request differently than it would have if the request came from an untrusted process.

Subjects could communicate with one another by using shared security-objects (managed by the kernel) such as shared memory segments, semaphores, etc. Each different mode of inter-subject communication was modeled as a separate kind of security-object. These inter-process communication (IPC) objects were storage objects just as were more persistent objects (e.g., directories and files), but were designed for more efficiency in IPC.

These architectures mapped well with security models such as the Bell and LaPadula model and definitions of subjects and objects corresponded closely with the definitions in the TCSEC.

1.2 ... And Basic Subject/Object Definition Now

In a microkernel-based trusted system, only the most basic system functions are implemented by software executing in the privileged hardware state. The remaining trusted system functionality is implemented by a collection of servers each of which executes as a process². Many microkernels, including the Mach microkernel which is the basis of TMach, do not provide sufficient functionality to implement subjects or security-objects. The server TCB (the portion of the TCB exclusive of the microkernel) uses the microkernel's basic services to construct subjects and security-objects.

This extensibility approach is enabled by the separation of the traditional kernel TCB into a microkernel and servers. Extensibility can be structured by a type mechanism. In a type-based client/server system, each server defines an abstract data type with a specific set of operations (or methods) defined for objects of that type. Each server is the manager for all objects of the type(s) it manages. In order to use a service based on some type, a client contacts the server that manages that type and sends requests to the server; each request is for an operation on some object of that type. Within such a type framework, extensibility takes the form of the definition of a new type and the addition of a server to manage objects of that type. As a result of using this framework, clients interact with new servers in the same way that they interact with existing system servers³.

Such a type-based approach to server extensibility provides a convenient framework within which to model the subjects and objects of a trusted system. The TCB is separated into a microkernel and a set of trusted servers which manage some fixed set of types. Adding new types and new servers must be modeled in such a way that the extension, while adding untrusted type manager servers, nevertheless does not change the subject/object model which defines the basic approach to the security of the system. In other words, new types of objects (managed by untrusted servers) can *not* be new kinds of security-objects.

There are a number of subject/object modeling issues that must be addressed within a microkernel/server architecture with type-based extensibility.

Trusted Servers are not subjects. The TCB servers are analogous to a kernel's process subsystem which implements subjects by associating processes with user IDs defined by an authentication subsystem. Therefore, the definition of subjects must carefully distinguish subject processes from the TCB process that implement

²In this context, we use the term *process* in a general way, to denote a domain of execution that is protected by the kernel yet is separate from the kernel's privileged domain.

³This form of type extensibility is increasingly referred to as "object-oriented" [9]. The underlying concept of object oriented design is that software is modeled as collections of cooperating objects. Object managers provide services in response to messages from clients or from other object managers. In order for trusted systems technology to keep up with this new approach it will be necessary for trusted systems design to be extended to meet this evolution in software design, development and analysis.

subjects.

TCB servers implement security-objects. When microkernel resources alone do not constitute security-objects, the server TCB must build on microkernel resources to construct security-objects. The microkernel provides primitive storage abstractions (e.g., memory and devices) which servers use to implement system objects such as files. Therefore the definition of security-objects must be enhanced to account for security-objects being managed not by the kernel but by TCB servers.

Both kernel and trusted servers offer the TCB interface. The servers use the kernel interface to implement objects. However, the kernel interface is not hidden by the interface that the TCB servers offer. Because kernel services are available to subjects, subjects can use the kernel services in exactly the same ways as TCB servers, i.e., to implement objects. In other words, subjects can be non-TCB servers and can manage objects. However, the set of kernel services available to subjects is restricted to a subset which has been determined to be non-security-critical. Other, privileged, kernel operations are restricted for the use of the TCB servers, and cannot be directly accessed by untrusted servers. Instead, untrusted servers call on the TCB to gain TCB-mediated access to resources governed by kernel privileges.

Microkernel provides basic TCB interface mechanism. The existence of TCB servers also effects the basic interface between subjects and the TCB. As in a kernelized system, subject processes trap into the microkernel. Then, rather than always servicing the request in the kernel—as is done in a kernelized system—the microkernel redirects some service requests to the appropriate component of the server TCB. Among these server-implemented requests are both subjects' requests for access to security-objects and also subjects' requests for access to subjects.

1.3 Where Do We Go From Here?

Each of these differences requires extensions to the traditional notions of subject, object and subject-TCB interactions. This paper describes one effort at such extension, performed as part of the development of Trusted Mach (TMach), a trusted system which has used TCSEC principles in the development of an extensible, type-based system.

This paper will first present an overview of the TMach system in Section 2. Then Section 3 presents the various issues pertaining to the application of TCSEC principles to TMach, including specific issues which motivate extending the TCSEC definitions to encompass microkernel-based client/server systems. Having laid this groundwork, Section 4 will then present an account of the way TMach addresses these issues by defining subjects and objects in a manner consistent with the TCSEC, and yet inclusive of the extensibility that is enabled by the microkernel-based, client/server architecture of TMach.

In making this presentation, subject-TCB interactions will be presented both from the point of view of subject-object interactions and subject-subject interactions. There are fundamental distinctions between these different views of TCB-interface usage, and these distinctions drive the different roles of three related but critically different system abstractions: the kernel's port abstraction, the server TCB's IPC objects, and a new kind of named object—the connection point—which is critical to modeling extensions of sets of types of objects. Each of these three will be described, including the role of each in TMach's provision of secure communication.

After the central presentation, Section 5 provides an analysis of the security of the TMach system given these new definitions and describes the benefits achieved from this new point of view. Section 6 then discusses extensions of TMach to a distributed system and points out the how the definitions of IPC objects contribute to ensuring security in a distributed system. Finally, Section 7 presents summaries and conclusions.

2 TMach System Overview

Trusted Mach is a microkernel-based system with a client/server architecture, which has been developed using an object-oriented design methodology. TMach is aimed at the B3 level of trust as specified in the

TCSEC and at the F-B3/E5 levels of the ITSEC. TMach is a trusted server software layer that runs on the Mach microkernel. The TMach servers use the Mach microkernel to implement security objects and subjects, to implement controls on the access of objects by subjects, and to implement mechanisms for supporting policies such as subject identification and authentication.

The TMach system uses a paradigm for computation known as a client/server architecture. In the paradigm, *server* processes provide services that are required by other processes, called *clients*, which request services from servers. The client/server interaction is via a form of message-passing. A client requests a service by sending a message to the server. The server performs the computation necessary for the request, and sends back to the client a reply message which contains the results of the computation. For example, when a TMach client requires access to data in a directory, the client sends a message to a TMach server component, which obtains the requested data and sends it in the reply message. The use of message passing as the means of client/server communication also facilitates distribution of processing.

The message-passing communication is provided by the Mach microkernel, which is the basis of the TMach TCB. The microkernel provides the primitive services that the server TCB uses to construct subjects, security-objects, the services based on interactions between them, and the access controls on those interactions. This kernel/server architecture is illustrated in Figure 1. We first describe the kernel's primitive services and then describe the various illustrated server components built on the kernel. We then discuss the security mechanisms of the kernel and how the servers use them to implement subjects and security-objects.

2.1 Kernel Primitives

The microkernel provides an active process-like system abstraction, three passive container-like system abstractions, and one primal mechanism—the port—that is interface to all abstractions and the microkernel services provided through them. These five kinds of abstractions, or *kernel-objects*, are: multi-threaded processes called *tasks*; *threads* of execution within tasks; regions of memory called *memory-objects*; devices; and message queues.

For each of these five kinds of kernel-object there is a descriptor called a port. Ports have capabilities called *port rights*. Possession of a *send right* to a port allows the possessing task to send messages over the port. In the case of ports which are descriptors for tasks, threads, memory objects, or devices, the microkernel receives each message and interprets it as a service request on the kernel-object to which the port refers. In the case of ports which are descriptors for message queues, the microkernel enqueues each sent message, which may later be dequeued by a task which holds the *receive right* for the port.

Ports, and messages sent on them, are the fundamental interface between tasks and the microkernel. Most microkernel interface functions are operations on one kind of kernel-object, and these operations are performed using the port that is the descriptor for the object. In many cases, the operation is implemented as a message sent by the task on the descriptor port. For example, there is an interface for mapping a memory-object, and this interface is implemented as a message sent on the port that is the descriptor for the memory-object. The message-send operation is implemented via a trap mechanism.

Just as the port is the kernel's fundamental mechanism, part of the port mechanism is the kernel's primary protection mechanism. Ports are used by name (actually an integer), but each task has a port name space which is mapped by the microkernel in a manner analogous to virtual memory. A task may attempt to use a port name, but the attempt will only be valid if the port name maps to an actual port. There will only be a valid mapping if the task has obtained a right to the port. A task can only acquire a port right if the right was contained in a message that the task received.⁴ Such acquisition occurs as a result of the following sequence of events. Initially there is a port A which is a descriptor for a message queue. Task R has the receive right for port A, and task S has a send right for port A. There is also a port B which task S has a send right for. Then task T sends a message over port A, and includes a port right for port B; task R

⁴Actually, this is a simplification—there are a few other ways that tasks can acquire port rights—but a useful one since the other methods also either involve the microkernel directly (a task can request that the microkernel create a new port and give the task a port right for it) or also involve the use of other port rights (if a task acquires rights to a port that is a descriptor for another task, the first task can get port rights from the second task).

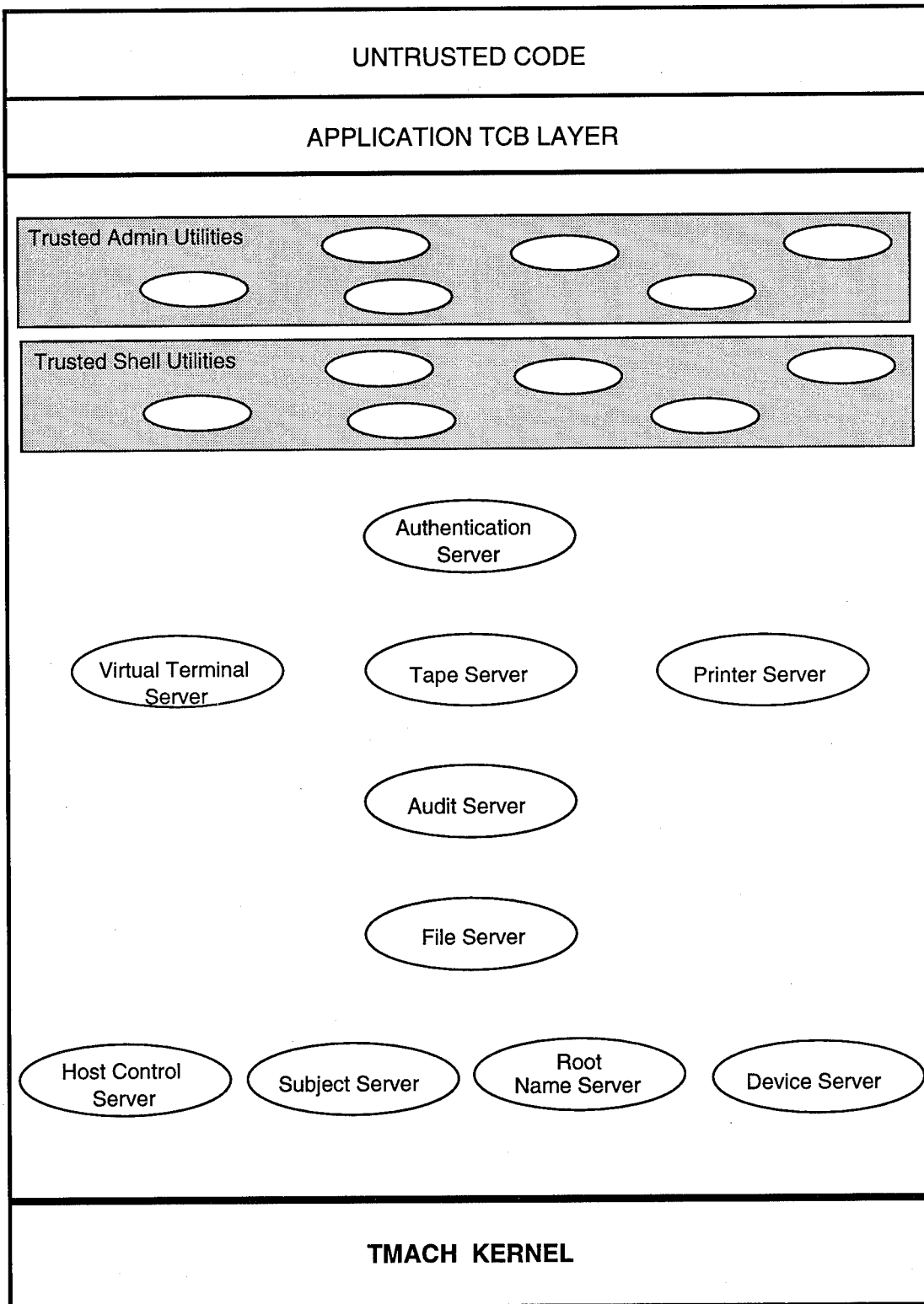


Figure 1: Trusted Mach Kernel/Server Architecture

receives the message, and the kernel updates R's port name space to have an entry for port B; task R now has a send right for port B.

Because of the microkernel's port name mapping, the port mechanism has two powerful but simple consequences. First, control over a task's set of ports is equivalent to control over the task's access to system resources. Second, possession of a right to any one port has the potential use of gaining other port rights, if any other tasks have a right to the same initial port and are cooperative in using it to send messages containing other port rights.

2.2 TMach Server TCB

As shown in Figure 1, the server TCB is comprised of a number of trusted servers and some utilities. In addition to using the kernel abstractions to build higher level abstractions, the TMach TCB servers also use the abstractions to protect themselves from each other and from non-TCB code.

Access mediation in the TMach system is centralized in the Root Name Server (RNS). All named entities are items in the TMach name space. The RNS manages the name space and holds all security-relevant information about the named items. The RNS makes all mediation decisions based on the TMach system security policy. While all the security-relevant information about an item is held by the RNS, the specific semantics of an item are implemented, and the contents held, by a different server called an item manager. There are several types of named items in the TMach system. Currently TMach provides trusted item managers or servers for directories, files, and various multilevel devices. In addition to item managers, audit and authentication services are each provided by a separate server. The other TCB servers shown in the architecture diagram provide specific services related to privileged kernel operations: the host control server for management of hardware configuration; the device server for management of physical devices; and the subject server for creation of tasks with arbitrary security ID's.

In addition, the TCB contains entities which are not actually servers but collections of programs with a common purpose. Specifically the Trusted Shell Utilities (TSH) and the Trusted Administrator Shell Utilities (TASH) are collections of programs used to configure and administer the TMach system.

The final layer in the TMach architecture is composed of the Non-TCB software. This layer provides the user-level interface, also called the operating system (OS) personalities. These non-TCB servers use the personality-neutral TCB servers to provide their own OS-specific services. Currently two specific OS personalities (POSIX and DOS/Windows) are being designed for the TMach system, but any number of other personalities are possible.

2.3 Kernel Security Features

The microkernel does not provide sufficient functionality for kernel-objects to be security-objects and subjects. None of the kernel-objects has any security attributes (e.g. ACL, sensitivity label, integrity class), so there is no basis for access control. With respect to security-objects, it is entirely up to the server TCB to build on kernel-objects, both by using them to construct higher-level abstractions which include security attributes and by using these attributes to implement access controls. The server TCB uses the kernel privilege of physical device access to securely store object data including security attributes.

With respect to subjects, however, the task kernel-object has two features which assist the server TCB in basing subjects on tasks. These two features of Mach tasks are process isolation and the security identifier. The Mach task is a familiar process-like abstraction, each task having a distinct virtual address space. The kernel uses privileged hardware features to implement virtual memory and to prevent any task from tampering with other tasks' virtual memory. In addition, each Mach task has its own protected port name space. As a result of using the familiar mechanisms that underly subject definition, Mach implements process isolation and extends the notion to include the management not only of memory but of ports, which are the critical access mechanism for all system resources.

The security identifier (or secID for short) is the second microkernel mechanism that supports the server

TCB's implementation of subjects. The secID is an attribute of each task. The microkernel provides the secID as an uninterpreted value which is intended for server-level use.⁵ Servers can interpret secIDs in whatever manner is useful at the server level. The microkernel merely maintains an immutable task-to-secID mapping, which is set during task creation by the creating task for the created task. In addition to this task-secID association, the microkernel performs one other function which relates secIDs to messages that are enqueued on message queues: when a task sends a message on a port for a message queue, the kernel stores the sender's secID along with the message; then, when a task receives the message, it can determine the secID of the sender.

This sender-secID tagging is critical for security in any Mach system. In Mach, all non-kernel computation takes place in tasks, and all interaction between tasks is accomplished by means of ports. Therefore, ports are the means of interaction between subject tasks and the tasks that comprise the server TCB. As a result, sender-secID tagging enables the server TCB to establish the identity of subjects, as described further below.

2.4 Subject/Object Abstractions

TMach subjects are based on Mach tasks. A subject task is created by the TMach server TCB when a logged in authenticated user requests creation of a session⁶. The server TCB calls on the microkernel to create the task and specifies the secID. The value of the secID is a token that represents the various security attributes of the subject: a user identity, groups, sensitivity level, etc. The ability to thus specify the secID of a child task stems from a kernel privilege⁷ which is held solely by the server TCB. Because subject tasks lack this privilege, any child tasks inherit the secID from the parent task. Thus, for each session, there is a task or group of tasks all with the same secID corresponding to the user.

The secID is used by the TMach server TCB to mediate and enforce access control decisions. This usage is based on the kernel's service of tagging each message with the secID of the sending task. Mach messaging is the interface between any task and the kernel, and between any task and other tasks, including the tasks that comprise the server TCB. Thus, any subject task request of the TCB is a message tagged with the subject's secID.

TMach security-objects are implemented by the TMach server TCB, using Mach devices, message queues, and memory-objects. Each TMach security-object has a name and a type. The name represents an item in the TMach name space, which is implemented by the Root Name Server (RNS). The RNS maintains a mapping between each item and its type, and each type and its item manager.

All named security objects are items of a type managed by a TCB server which acts as a trusted item manager for the type. Each TCB-managed type is a specific kind of security-object. Within the TMach hierarchical name space, all internal nodes are *directory* security-objects. Each external, or "leaf", node is an item of one of several types. *File* security-objects provide general purpose storage. *Symbolic link* security-objects provide pointers within the name space. *Type* security-objects are items which describe a type associated with some other items. There are various types of *device* security-objects which are implemented by the server TCB's use of device kernel-objects coupled with server-implemented access controls. There may also be other types, types which are not managed by the TCB. Every item of any non-TCB-managed type is a security-object called a *connection point*. Finally, there is one kind of TMach security-object which is unnamed: the *IPC object*. The core of this paper is Section 4's description of connection points, IPC objects, subjects and non-TCB item managers.

In order to gain access to a named security-object, a subject sends a request message to the server TCB. The RNS is the destination of all such messages. The RNS performs all access mediation and is the sole source of access to every security object. The request includes the name of an item and the access mode

⁵The kernel itself uses secIDs in one way, to enforce access-control decisions on memory-objects, in the same manner that server-level item managers do (see below). This checks does not involve interpretation of the secID, but rather is a check on equivalence of secIDs.

⁶A user can create multiple sessions each potentially of a different security level.

⁷Kernel privileges are represented as ports. For a task to successfully call a privileged kernel interface, the task must have a port right for the port representing the privilege appropriate to the interface.

requested. The RNS “resolves” the name, i.e., determines which specific item is named and retrieves from stable storage the attributes of the item. Among these attributes are security attributes, e.g., a label and an ACL, that are half of the input to the access mediation function. The other half are the security attributes of the subject requesting access. These are obtained from the secID tag on the request message. The RNS extracts the message secID, expands it to the corresponding full set of attributes and uses these subject attributes to decide whether to grant access in the requested mode.

If access is approved, the RNS creates a message queue. The port of the message queue is used as a descriptor for the subject’s access to the requested object. The message queue itself is used as the communication medium between the subject task and the item manager. The RNS gives the port’s receive right to the item manager and gives a send right to the subject task. As a result, the subject task and item manager can then interact in client and server roles, because the the subject task (client) can send operation requests to item manger (server).

Trusted item managers have a security function that is also dependent on the secID functionality of the kernel. After access to an item is granted, the item manager receives operation requests on the item descriptor port. However, because rights to the port can be passed among tasks, the item manager checks the operation request message and honors it only if it originated from the subject to which the RNS granted access. This check is a comparison of the secID of the original access requester and the secID tag of the operation request message. A similar check is made between the access mode required for the operation and the mode of access granted by the RNS: when the RNS sends the item descriptor port right to the manager it also includes the requester’s secID and access mode.

Therefore, item managers enforce the rules that a user task with a send right to an item descriptor port can only use that port to access an object if the task has the securityID prescribed by the RNS, and only if the requested operation’s access mode was granted by the RNS. The RNS’s mediation and the item managers’ enforcement are the central mechanisms of access control in TMach.

Note that multiple tasks can share a secID. As described above, a user’s original task can spawn child tasks with the same secID and hence the same single sensitivity label. All of these tasks form a *task group*. When access is granted to one task in a task group other tasks in the task group can use the access: the original accesser can send an item descriptor port right to another task in the task group, the other task can use the port to send an operation request message, and the item manager will honor the request because it has the correct secID. Thus, TMach’s subject is the task group. Because each session is assigned a distinct secID, each session’s task group is a distinct subject.

This feature of TMach’s subject definition and access control mechanisms allows for a potentially powerful multi-programming approach to application development. Client applications can use multiple co-operating tasks, rather than being required to either have the entire application reside in one address space, or have separate tasks get separate access to shared objects.

3 Issues in Applying TCSEC

Having presented a basic picture of the subjects and objects of TMach, it should be clear that a new view is required to provide a more complete and detailed picture. The new view of modeling subjects and objects in TMach is an extension of the TCSEC view of subjects and objects as embodied by many of the early systems, which many view as definitive implementations of the TCSEC concepts pertaining to modeling of subjects and objects. We have extended the TCSEC view to include a trusted system that has been designed for type extensibility mechanisms which allow the system to be extend without changing the TCB and its interface. Before describing this extended view and completing the picture of TMach subjects and objects, there are a few important distinctions to be made. These distinctions are between the various kinds of interfaces in TMach.

In a kernelized TCB, the main kind of interface is the kernel’s interface to subjects. This interaction is also present in TMach, as the microkernel’s interface to tasks, both to subject tasks and to TCB tasks. However, because of the microkernel/server distinction, the TCB interface consists not only of the kernel interface,

but also of the interface between subjects and the server TCB. The server TCB interface is message-based, using the microkernel mechanism of message queues that can be shared by tasks, e.g., a subject task and a TCB server task.

This same message mechanism is also used for a third major interface, the interface between a subject and another subject. The principle distinction of the subject-subject interface is that the server TCB *mediates* connections between subjects.

Because the server TCB mediates subjects' use of the kernel's communication mechanism, subjects' interface to one another is via a server TCB interface for requesting access to a communication medium to another subject. This request, being the same sort of request as that to access objects—and having the same kind of mediation—is in essence a request by a subject to access a subject. However, to avoid modeling a subject as an object, there is a new security-object, the *IPC object* which represents the set of resources of one subject that another subject can access. From a high-level conceptual view each IPC object is the passive portion of a subject. The next section gives more details on the use of IPC objects to model access of one subject to another subject's resources.

In an extensible client/server system, perhaps the most significant modeling concepts center around modeling subject-subject communication. Connection points are used to model such interaction in an extensible system. The relationship between the two kinds of security objects, connection points and IPC objects, is the main topic of the remainder of this paper. A key concept in this relationship is the *communicating group*. A communicating group is a group of subjects that can communicate among one another. Each subject initially is alone in a communicating group, but through TCB-mediated access requests the subject can get in communication with another subject. As a result the two subjects become part of one communicating group. If either or both subjects were previously in communication with other subjects (i.e., were part of a larger communicating group), then all of these other subjects are also part of the newly merged communicating group. This transitive group membership is intended to model the fact that when one subject communicates with two other subjects, each of those other two is potentially in communication with the other via port rights that the first subject could pass to the other two.

The next section uses these concepts to present a new view of subjects and objects that can account for subject-subject interactions, specifically those which are interactions between a client subject and a server subject which implements objects that are not security objects. This situation arises when the system is extended with new types of objects and managers for them. Because all processing occurs in the framework of client requests to object managers, it is possible to develop a model of subject/object interaction which can be extended to include interactions between untrusted applications and objects. Constructing such a model of communication has allowed us to extend the abstract concepts of subject and object closer to the application level. We believe that this communication model will allow application designers to close the gap between minimal TCB security primitives and more complex application needs, and to do so in a way which can be shown to be secure.

4 A New Point of View

Given the basic definition of subjects and objects in TMach, the overall picture must be rounded out by consideration of two related questions, which concern the areas of TMach that are modeled most differently from early TCBs. These questions are: How are subject-subject interactions modeled? How are subject-object interactions modeled, when the object is an item that is not managed by the TCB? These questions are closely related, because the only means of interaction between subjects is via non-TCB-managed items.

Before describing the details of subject-subject interaction, however, we must first understand the initial state of a subject, and the TCB's controls over transitions from the initial state.

4.1 Initial State

Initially, a TMach subject is simply a task which cannot contact any other subjects and is not accessing any security-object⁸. There are two types of action an initial subject can take in order to use any other system resources than the ones it was created with. First, a subject can always make microkernel calls. Secondly, a subject can contact the server TCB via the kernel's message-passing service.

By contacting the microkernel, the subject can gain more primitive resources. However, doing so only adds to the primitive resources that comprise the subject. These additions are examples of the subject's modification of its own IPC object. However such IPC object access does not permit any new object access or subject contact. The following are examples of a subject modifying its IPC object: a subject can call on the microkernel to create other tasks or threads which become part of the same subject; a subject can call on the microkernel to create memory-objects, which adds to the virtual address space of the subject; a subject can call on the microkernel to create message queues, though with no other subjects with rights to the messages queues these are little more than extensions to the subject's address space; finally, a subject could call on the microkernel to access a device, but such requests would be disallowed because device access requires a privilege that the server TCB holds and does not give away to subjects.

Other than these kernel interactions, any subject activity must come about as a result of requests to the server TCB. Each subject initially has only one port right, a send right to a port for a message queue.⁹ The TMach Root Name Server holds a receive right for this port. Therefore the port can be used by the subject to send request messages to the TCB. As described in Section 2, the Root Name Server (RNS) is the sole point of access for all objects, and a successful access request results in the client acquiring a send right to a descriptor port for the requested object. Because all system resources are accessed by subjects in the form of security-objects (with the above-described exception of kernel resources which only augment the subject itself), these object descriptor ports are the sole means of access to resources.

Object access is only possible via the RNS. Therefore, the RNS, together with the managers of the objects to which the RNS has granted access to a subject, has the ability to control accesses by that subject. However, correct control depends on the correct management by the trusted system servers of the ports to which a subject is given rights. The rules for correct control can be simply stated. First, there is a rule of correct access granting: for the ports over which the RNS receives messages, the RNS only handles object request messages, and only replies with object descriptor ports when access control checks were successful. No other requests are honored, and no other ports are given to subjects¹⁰. Second, there is a rule for correct continuing access: trusted object managers only use item descriptor ports for providing access to the single object that the port is the descriptor for, and access is only provided if the requester is the same subject that originally opened the object.

Thus far, we can see that a subject may open objects and get descriptor ports, but that these ports are only useful for communicating with the TCB, i.e., the trusted servers that manage the security objects which the subject is accessing. Using the above mechanisms for communication with the RNS and trusted object managers, a subject never acquires any port rights that will allow it to communicate with another subject. However, the TCB does provide a way for subjects to contact one another. Such contact involves an object that is of a type that is not managed by the TCB. Because TMach's type system is extensible, it is possible for a subject to define a new type and to become the manager for that type. Then, when another subject requests access to an object of the new type, two port-related actions occur: the requesting subject receives the usual send right to the descriptor port; and an untrusted object manager receives the receive right to the descriptor port. There is a critical distinction in this mechanism in that the object manager is a subject (rather than a TCB component), the object manager and requester are two subjects in communication with each other via the port to which they share rights.

⁸Actually the subject is created with access to its own internal state which is modeled as an IPC object.

⁹Actually, a task does have other port rights (such as for the port that is the descriptor for the task itself), but these do not effect the subjects' ability to contact other subjects.

¹⁰The RNS is also an object manager for some types, e.g., directories. In this context, we distinguish between the RNS—the central point of access and mediation—from the object manager for directories. The object manager components of the RNS are treated in exactly the same way as object managers that are separate servers.

4.2 Connection Points

However, this subject-subject communication raises some issues that must be addressed. Recall that each TCB-managed item is of a type that corresponds to one kind of security-object, e.g., a file item is a file security-object. However, what kind of security-object is an item of a non-TCB-managed type? A TMach system could be extended to have several non-TCB-managed types, e.g., mailbox, calendar, database. However, *none* of these types is a new kind of security-object. From the application point of view of the system, an item of some new type (e.g., calendar) is not fundamentally different from an item of system type, e.g., file. From the TCB definition point of view, however, there is a critical difference: an untrusted item manager cannot be relied upon to correctly implement the rule for correct continuing access described above in Section 2.4.

For example, there could be two non-TCB-managed items of the same type, one with an ACL that only allows reading and writing by one user, and another object with an ACL that only allows reading and writing by another user. When a client opens the first item for write, the RNS will check the ACL to ensure that writing is only allowed by the authorized user, and only then is the untrusted item manager involved. When a client opens the second item for read, a similar procedure is followed. However, when the second client does a read request, the untrusted item manager is free to return data that was previously written on the first item, to which the second client is denied access by the ACL.

Clearly, such misbehavior is not desirable for a useful item manager that operates as expected in the type-based client/server framework. However, the critical point for TCB definition is that the TCB must assume that such misbehavior is possible.

At this point, we are now ready to address the question of modeling subject-object interactions for non-TCB-managed items. For purposes of TCB subject/object definition, all items of all non-TCB-managed types are considered to be security-objects of one kind: connection point. Each connection point object is simply an item which can be opened by a subject for the purpose of communicating with another subject. For each non-TCB-managed type, all items of that type are connection point security-objects, but each is a different name for the capability to contact the same subject, the type's item manager.

Connection points are different in one important way from all other named objects. All named security objects including connection points have in common data such as ACL, label, modification date and time. In addition each other security object of TCB managed type has type specific data; for example, a file has file contents; a type object contains data about operations and access mode. However, connection point objects contain no further data. Connection points are security objects which model items that are not managed by the TCB. Thus, from a modeling point of view there is nothing more to be said about these objects. Yet, from an application point of view these type specific contents of the object are managed by the untrusted item manager.

4.3 IPC Objects

Now that we have described the mechanism for subject-subject communication (non-TCB-managed items) and also explained its consequences for modeling objects (connection points), we can complete the account of TMach's subject and object definitions by considering how to model these subject-subject interactions. The TCSEC paradigm does not allow for direct subject-subject interactions. Rather, subject-subject interactions are modeled by means of some intervening object. Therefore, in TMach, subject-subject interactions are modeled in terms of a kind of security-object called an *IPC object*.

As mentioned in Section 3, the IPC object is used to model the passive part of a subject, its state, which another subject can access. More specifically, an IPC object is the sum of the states of all the tasks that comprise a subject. The state of each task is the set of microkernel-objects it can use: memory objects, message queues, threads, and tasks; each of these is represented by a port to which the task has a right. In addition to ports, the other part of a task's state is its virtual address space, which allows a task to access memory directly without using a port as a descriptor.¹¹ Therefore, in terms of kernel mechanisms,

¹¹Memory is the exception to the rule that all kernel resources are accessed via a port. Once a memory object has been mapped

the content of an IPC object is a set of memory regions and a set of port rights, each of which is a descriptor for a kernel-object accessible to some task in the subject to which the IPC object corresponds. With regard to subjects, each IPC object represents the whole of one subject's operational environment—essentially its virtual address space and port name space—that can be effected by another subject.

Because IPC objects are security-objects of TMach, the definition of security-objects must include an account of the mechanisms and modes of access to IPC objects. All other kinds of TMach security-objects are accessed initially by opening the named item that corresponds to the security-object; subsequent access is via the client/item-manager interface of the type of the item. Creation and deletion are also accomplished via item open and management interfaces.

IPC objects, however, use different mechanisms than named security-objects. All IPC object operations (create, access, delete) are side-effects of other operations. No IPC-object operations are undertaken by reference to the IPC-object itself; in fact, there is no name by which to reference an IPC-object. An IPC object is created each time a subject is created. The IPC object is destroyed along with subject, i.e., with the destruction of the last task of the subject. There is only one access mode for IPC objects: all accesses permit arbitrary use and modification of the IPC object.

Although such arbitrary access is possible in principle, the access is in fact constrained by the subject associated with the accessed IPC object. If the accessed subject is willing to pass all its port rights to the accessing subject, then complete access will be possible. On the other hand, if the accessed subject passes no further port rights, then access will be limited to sharing the message queue represented by the port right, the sharing of which was established during an open of a connection point.

There are two ways that access to an IPC object is granted. In the first case, each subject is granted access to its associated IPC object when the subject is created. Subsequently, the subject can access the IPC object in a variety of ways, by accessing memory or any kernel-objects to which a subject's tasks have access, or by creating kernel-objects, or by destroying any kernel-objects to which the subject's tasks have access.

The second method of IPC object access occurs when a task of one subject acquires a port right to a port for which another task (of another subject) already has a port right. The canonical example occurs during an open operation on a connection point. As described in Section 4.1, rights to the same port are given to both the opening client and the item manager associated with the connection point. Thus the subject *M* (of which the item manager task is a part) acquires access to the IPC object associated with the subject of which the client task is a part, similarly for the client's subject and *M*'s IPC object.

Given this initial access, each subject has the discretion to expand the amount of accessible resources by passing further port rights in addition to the single initially shared port. Because the initial port sharing is mediated by the TCB, and because any subsequent additional port sharing is discretionary, we can see that IPC objects model the two salient features of inter-subject interaction in TMach: first, any established communication has the potential to be expanded beyond the original shared port; second, every original port sharing is mediated by the TCB, and mediated according to a policy that includes for the possibility of such expansion.

This expansion of access is related to the transitive nature of IPC object access. We have already seen that the canonical method of IPC object access is via an open of a connection point, when the client and manager obtain access to each other's IPC objects. All other IPC object accesses also occur during connection point opening, and these other kinds of access are transitive. To understand the transitive nature of IPC object access, recall that an item manager can have potentially several clients. Each client, furthermore, may be in contact with other managers, and so forth.

The complete network of inter-communicating subjects forms a communicating group. Each time a connection point access is approved by the TCB, the communicating group of the client is merged with the communicating group of the manager. The client's and manager's subjects gains access to each other's IPC

into a task's virtual address space (an operation that uses the memory object's descriptor port), the task has the usual sort of virtual memory-mapped access to the region of memory associated with the memory object. Direct memory access is of course necessary in practice, but can be considered an optimization of memory object read and write operations which require use of the memory object's descriptor port.

object; but every subject in client's communicating group gains transitive access to the IPC object of every subject in the manager's communicating group, and vice versa. As a result, every subject in the merged communicating group has access to the IPC object of every other subject. This transitive closure of access models the *potential* of every task to share all its port rights with every task it communicates with, and for those tasks to further pass on the port rights.

With regard to security, the critical point is that each IPC object access occurs during communicating group merger or subject creation, which is performed by the TCB only after passing access control checks for a connection point. Mandatory security is maintained by ensuring that a subject can only join a communicating group comprised of subjects of the same level. Therefore the relationship between IPC objects and connection points can be summarized as follows: for each non-TCB managed type, all items of that type are connection point security objects, each a different name for the capability to access both the type's item manager's IPC object and all the IPC objects of the subject in the item manager's communicating group.

5 Security Considerations

The above sections have described the subjects and security-objects of TMach and the server TCB's use of kernel mechanisms to implement them. This section concerns the *assurance* that the system enforces its security policy. Formal assurance is addressed by a formal model [6] of the system's entities and rules of operation. After summarizing the model entities, this section addresses design assurance by describing how TMach implements unbyassable security mechanisms within a microkernel-based client/server architecture. Next we address architectural assurance by discussing how this architecture has benefits that enhance the assurance of the system's implementation of the security mechanisms.

The TMach subject is a set of tasks of one session, which therefore have the same mandatory and discretionary security attributes. The server TCB encodes these attributes in a token which is stored by the microkernel in each task's security ID attribute. TMach implements several kinds of named security-objects. The names are derived from a hierarchical name space. Each item in the name space has type. Some types are managed by the TCB: directories, files, symbolic links, types and various types of devices. One kind of security-object models each of these types. One other kind of security-object, the connection point, models all other types, i.e., those that are not managed by the TCB. A subject's access to all named security objects is mediated based on the subject's security attributes (encoded in the security ID which the kernel affixes to every subject's requests) and the object's security attributes (maintained by the Root Name Server). The remaining security-object, the IPC object, models the interactions between subjects that result from clients accessing items of non-TCB-managed types. IPC object access is a side-effect of connection-point access.

5.1 Unbyassable

The unbyassability of the TMach TCB is built up from hardware mechanisms, kernel services implemented using those mechanisms, and server TCB access control functionality built on kernel services. Because only TCB software runs in the most privileged hardware state, it has sole access to hardware resources and services, including sole control of physical memory. The kernel uses these hardware mechanisms to protect itself and to implement the virtual memory mapping and port name mapping mechanisms that protect tasks from one another.

The RNS is the server TCB component that mediates access. The RNS runs not in the kernel's privileged hardware state, but in a task. Therefore, in addition to the demonstration of the kernel's unbyassability due to hardware use, there must be a higher-level demonstration of the RNS's unbyassability, i.e., that subjects may obtain access to resources only after appropriate mediation by the RNS. Each TMach subject has a well-defined initial set of resources and available services, this initial state ensuring that additional resources may only be obtained after RNS mediation.

A subject's use of microkernel interfaces is also an issues with unbyassability. As described in Section 4, subjects initially have only two capabilities: communication with the kernel via the hardware trap mechanism

and use of the kernel port mechanism to communicate with the RNS. Therefore, a subject initially can contact no other task than the RNS. Subsequently, subjects obtain resources only after RNS mediation. Subjects can, nevertheless, contact the microkernel without going through the RNS. However, microkernel services cannot be used to access objects without RNS involvement. Demonstration of this point corresponds to the services of the kernel. A subject can manipulate its IPC object (create child tasks with the same security ID, manipulate its threads, or create new threads, use existing or create new ports and memory objects) but these operations effect the state of the subject, but do not effect any other resources mediated by the RNS. The remaining kernel service is for devices, but the direct access to devices is controlled by a kernel privilege which the TMach TCB reserves for its own use. In TMach, subjects do not have this privilege and hence cannot obtain device access from the kernel.

The microkernel provides three privileges, each represented by a port. In order for privilege operations to succeed the caller must have ports rights to the appropriate port. The first privilege is to devices, represented by the device port; next is host control represented by the host control port; and the third is the ability to create tasks with arbitrary security ids, which is represented by the host security port.

Because of TMach's multi-server architecture, communication between the servers is also a critical part of the basic security mechanisms. TCB servers must be able to accurately identify one another. For example, item managers enforce access decisions communicated by the RNS, so item managers must be able to ensure that such access directives genuinely come from the RNS. The microkernel's port mechanism provides such identification. During bootstrap, TCB server tasks are created with rights to ports shared only by TCB servers. By only using these ports and by never passing rights to them, TCB servers ensure the authenticity of other servers.

The kernel's secID service is also the foundation of access control. Subjects can only contact the server TCB via the kernel's IPC service. Therefore, all subject requests are tagged with secIDs by the kernel, and the TCB servers use the secID for access control. The other key mechanism for access control is the server TCB's storage of item security attributes, and of type specific information specifying for each operation what access mode(s) are required. These security-critical data are TCB internal data inaccessible to subjects.

All these fundamental mechanisms extend simply when the system is extended with untrusted item managers. As with all items, the server TCB maintains object security attributes, whether or not the item manager is trusted. Operations on items of non-TCB type are treated by the TCB as always requiring read-write access, because access to non-TCB-managed items is really access to the subject that is the item manager. Access to the subject allows arbitrary communication with that subject, so RW access modes are needed to ensure that only subjects of the same label can communicate.

Of course, an untrusted item manager *may* correctly implement a type and operations on its objects. Each item *may* be correctly implemented as a distinct object with its own distinct content. The TCB cannot assume this, and this is the reason for treating non-TCB-managed item access as subject access rather than object access. However, if an untrusted item manager does correctly implement the access control mechanisms that trusted item managers do, then its objects will be appear to applications to be very similar to the objects of the TCB.

5.2 Benefits

The TMach system design is based on extensive use of *layering*, *modularity*, *abstraction* and *data hiding*. Layering increases assurance by dividing the system into a collection of layers, from the most primitive layers to the highest or least primitive layers. Within each of the abstract layers of the system architecture (i.e., kernel, servers, OS personalities), each of the layers is further subdivided. Modularity increases assurance by grouping together like functions into design and implementation units. As with layering, there are several levels of refinement of modularity in TMach. First there are the layers of the system as a whole (non-TCB, Server TCB, microkernel TCB) then each of these is further decomposed into subsystems and these are decomposed into individual modules. The layering and modularity along with the object oriented design of TMach provides abstraction and data hiding by providing progressive levels of interfaces and services. The principle of least privilege also plays an important role in TMach. At the lowest layer each module

in TMach is designed to perform its intended function and no more. Further, modules and layers control the export of privileges and services to only those needed by higher layers which can be shown to be safely exported. Finally, through the use of domain separation between trusted servers and between servers and the microkernel, the concept of least privilege is enforced throughout the system. All of these combine to provide increased assurance that the system enforces its security policy.

The TMach architecture, with its extensible type-based client/server design, presents many advantages. As we have seen, the basic abstractions of subjects and objects have been carefully constructed to provide extensibility. Because of the extensive use of layering, modularity, abstraction and data hiding within the context of an extensible type-based model of operation, an untrusted server for some new type of item can be introduced without effecting the basic definition of subjects, objects and the rules for secure interactions between them.

Along with extensibility comes flexibility. All access mediation in TMach is performed by the RNS, and all mediation computation is performed by one module that compares subject and object security attributes. As a result it is possible to replace the RNS's mediation module with some other mediation module which performs different or additional security policy mediation. The particular security policy enforced by the RNS is independent of the client/server design and the basic rules guiding subject-object and subject-subject interactions.

Finally it is important to note that Mach, and correspondingly TMach, has been designed with portability as a goal. The Mach microkernel encapsulates all machine dependencies into a few specific subsystems. The microkernel is a machine dependent base which isolates non-kernel software from the idiosyncrasies of differing hardware bases. Only these specific microkernel subsystems require modifications for a new hardware base. This approach is essential for meeting the goal of portability. TMach server software has no knowledge of hardware features other than those provided by the the microkernel and, therefore, need not be modified when TMach is ported to new hardware. Because the security objects are constructed by the server TCB layer, rather than the microkernel, security object abstractions are also portable.

Flexibility and extensibility have benefits for re-evaluation. Conscious attention to transportability and expandability in a trusted system context will in itself make re-evaluation easier and provide greater assurance in the trustworthiness of the system by forcing a modular design with narrow, well defined interfaces

6 Application to a Distributed System

TMach's subject/object definition is easily adapted to fit a distributed trusted system. The Mach microkernel itself was designed for distributed functionality, with an approach ideal for high-assurance systems. The microkernel itself is minimal, and manages only local hardware-based resources; distribution functionality is handled by a separate component that runs in a server rather than being part of the microkernel. This additional server provides a distributed service with the same interface and functionality as the microkernel's IPC service based on ports and message queues. This *distributed IPC server* allows for messages to be sent between tasks on different hosts in a distributed system. Message senders and receivers use exactly the same port mechanism as with local IPC, and in fact they need not be aware whether other tasks are local or remote. This property is referred to as the *transparency* of IPC in a distributed environment.

Because the port is the interface to all Mach microkernel services, distributing the port mechanism is all that is required to distribute all microkernel services. Likewise, because all of TMach's named security-objects are accessed via item descriptor ports, distributed IPC also suffices to provide distributed access to named objects. As for subjects, access to subjects is via IPC objects, which consist of a set of ports; again, distributing the port services suffices to provide distributed access to IPC objects.

Other than ports, secIDs are the other security-relevant mechanism relevant to distribution. Because access control depends on interpretation of secIDs, each host in a distributed system must interpret each secID the same way. This can be accomplished by cooperation between the Subject Servers—the Subject Server is the TMach component that handles the mapping of secIDs to security attributes—of the various TMach nodes in a distributed system. Correct message secIDs also depend on remotely originating messages being

locally delivered with the secID of the actual remote sender, rather the secID of the distributed IPC server. Therefore, the distributed IPC server is the sole holder of a kernel privilege that allows it to set the message secID of messages it sends.

Server-to-server cooperation is based on communication via distributed IPC. Distributed IPC enables inter-host cooperation between various servers. For example, the set of Root Name Servers can cooperate to provide a global name space, and item managers can cooperate to provide object replication for high availability, fault tolerance, and locality. The Triad project is currently developing a distributed TMach system that combines these features with support for real-time applications.

7 Conclusions

This paper has presented an overview of the TMach system and a number of issues pertaining to the application of TCSEC principles to TMach, including specific issues which motivate extending the TCSEC definitions to encompass microkernel-based client/server systems. Having laid this groundwork, we then presented an account of the manner in which TMach addresses the issues by defining subjects and objects in a manner consistent with the TCSEC, and yet inclusive of the extensibility that is enabled by the microkernel-based, client/server architecture of TMach.

Modeling subjects and security-objects in the context of type extensible client/server architectures is a new and critical aspect of modern trusted system development. The model must describe the security features of the extensibility mechanisms. This report described one approach to extending the TCSEC modeling concepts to a trusted client/server system with secure extensibility that derives from the microkernel basis

As was discussed in Section 5.2, extending TCSEC modeling concepts to encompass a type-based client/server extensible system like TMach has many advantages. The primary advantage is a secure approach to adding application specific extensions through the use of new non-TCB servers. This approach rests its security on the modeling of subject to subject communication. While some of the modeling concepts may introduce a degree of complexity, we believe that the increased assurance to be gained from type based extensibility easily offsets this complexity.

As trusted client server architectures become more prevalent we believe that there will be increased need for abstract security modeling concepts which can encompass type based extensibility.

References

- [1] *Trusted Mach System Architecture*, TIS TMach Edoc-0001-93B, Trusted Information Systems, Inc., 24 May 1993.
- [2] Accatta, M., Baron, R., Bolosky, W., Golub, D., Rashid, R., Tevanian, A., and Young, M., *Mach: A New Kernel Foundation for UNIX*, Proceedings of USENIX, July 1986.
- [3] *Trusted Mach Philosophy of Protection—DRAFT*. Document No. TIS TMACH Edoc-0003-94A
- [4] Department of Defense Trusted Computer System Evaluation Criteria. Technical Report DOD 5200.28-STD, DoD, December 1985.
- [5] Information Technology Security Evaluation Criteria. Technical Report 1.2, Department of Trade and Industry, June 1991.
- [6] A Mathematical Model of TMach. Technical Report TIS TMACH Edoc-0017-93A, Trusted Information Systems, Inc., December 1993.
- [7] *Trusted Mach System Architecture*. Technical Report TIS TMACH Edoc-0001-94A, Trusted Information Systems, Inc., August 1994.
- [8] D. E. Bell and L. J. LaPadula. *Secure Computer Systems: Unified Exposition and Multics Interpretation*. Technical Report MTR-2997 Rev. 1, MITRE Corporation, Bedford, MA, 1976.

[9] Booch, Grady. Object Oriented Design With Applications. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1991.

THE NEW ALLIANCE: GAINING ON SECURITY INTEGRITY ASSURANCE

By

Rene' H. Sanchez
Rockwell Space Operations Company
AIS Security Engineering and Operations
600 Gemini, R11A
Houston, Texas 77058
713-282-4589 FAX: 713-282-4922
E-mail: rhsanche@rsoc.rockwell.com

and

Donald L. Evans
UNISYS,
Government Systems Group, Space Systems Division
Mission Operations Directorate AIS Security Engineering Team
600 Gemini, U06b
Houston, Texas 77058
713-282-4050 Fax 713-282-4575
EMAIL : dlevans@rsoc.rockwell.com

Introduction

As the complexity of today's distributed computing environments continues to evolve independently, with respect to geographical and technological barriers, the demand for a dynamic, synergistically integrated, and comprehensive automated information systems (AIS) security control methodologies increases. Such business environments have introduced significant opportunity for process reengineering, interdisciplinary synergism, increased productivity, profitability, and continuous improvement. With each introduction of a new information technology (IT), there exist the potential for an increased number of threats and vulnerabilities which together comprise total risk. This is the level of risk that a management team must consider as an added cost of doing business. These costs may therefore be realized in the form of systems failure and loss of critical data. And with respect to mission and/or life critical systems, these costs may be too great to recover. It is in this context that management enterprise teams continue to place greater demands for products and systems which are dynamic, synergistically integrated, and equipped with high integrity AIS.

This paper describes the business approach employed at the National Aeronautics and Space Administration's Johnson Space Center Mission Operations Directorate (NASA JSC MOD) for bridging the gaps between the three key area product development support

functions: configuration management, AIS security, and quality assurance organization. This approach forms an enterprise-wide alliance needed for assuring the integrity, reliability, and continuity of secure IT products and services. Although the development and maintenance concepts for high-integrity unclassified systems are specifically addressed, the processes described are equally applicable to classified systems.

MOD AIS Security Program Challenges

Change is not easy whenever an enterprise considers reengineering its business processes. This kind of competitive business initiative typically involves redesigning and retooling value added systems for new economies. The AIS Security Program at the NASA JSC MOD is charged with directing and managing the business of information security for life and mission critical systems associated with Space Shuttle and Space Station operations facilities.

These systems encompass:

- some 3,600 personnel,
- 1,682 large mainframes, mini mainframes, distributed systems,
- five types operating systems,
- and a variety of network and communication protocols.

Much of these are legacy systems and are being pulled along by new technology making change very difficult to manage in this complex environment. The speed at which new emerging information technology is introduced to market, has also made it difficult to maintain an automated information systems (AIS) security control architecture baseline. Continued budget considerations have become a recognized element in managing this change. The MOD AIS Security Program has dealt with these complicated challenges head-on so as to comply with OMB Circular 130-A and the Johnson Space Center (JSC) Automated Information Systems Security Manual 2410.11. To this end, several interesting findings have resulted during the development and implementation processes used for accrediting NASA JSC MOD data processing installations (DPI).

Security Control Architecture and Complimentary Tools

The Security Control Architecture (SCA) has been in development since 1992 and is the product of a lot of in depth contemplation, research, and hard work by members of the MOD Automated Information Systems Security Engineering Team (ASET) and Rockwell Space Operations Company (RSOC) Security Engineering and Operations team. The SCA was implemented as the tool of choice for accrediting NASA JSC MOD DPIs in response to NASA's budgetary constraints. The scope and intent of the SCA document is to develop a comprehensive security control baseline architecture for a target DPI. The security control baseline architecture considers all functional platforms of a target DPI including: host mainframes, workstation(s), servers, bridges/routers/gateways, front-end processors/cluster controllers, network analyzers, and physical security.

The Automated Information Systems Security Reference Structure (ASRS) was created to document all information technology (IT) security terms and definitions in one reference structure. A complete volume of standard operating procedures have been developed and approved for use to support the DPI accreditation process at NASA JSC MOD.

Security Integrity Engineering Process

In today's computing world, distributed processing technologies change faster than most operational platforms can be baselined. As they evolve with an ever-increasing speed, companies and agencies are challenged with an opportunity to maintain stability for growth and strategic competitiveness. Management must consider that sensitive business systems increasingly demand higher levels of integrity in system and data availability. Within this framework reliability, through product assurance and security assurance constructs, provides a common enterprise objective. Accordingly, the scope of an enterprise-wide product assurance partnership must be expanded to all three functional areas as a single, logical, integrated entity with fully matrixed management (i.e., both horizontal and vertical management control). The process in which requirements for new information technology are infused into the enterprise and managed becomes the pivotal business success factor that must be defined, disseminated, and understood by the key functional support organizations.

New Alliance Partnership Model (NAPM)

It has become critically essential for enterprise management to gain an understanding of the interdependencies and complimentary pursuits that exist between the Quality Assurance (QA), Configuration Management (CM), and the AIS Security Engineering organizational support functions. With this knowledge, it is equally important to identify and examine a synergistic approach for realizing additional economies(cost savings/avoidances) throughout the system development life-cycle with continuous improvement techniques.

Implementation of product assurance and secure information technology development is a management decision that must be judiciously exercised and integrated as part of a system control architecture. In this model, AIS security management is qualified as the functional point of control and authority for coordinating and guiding the development, implementation, maintenance, and proceduralization of information security into a unique, integrated management team. The SCA is the approved strategic methodology used to produce a composite system of security controls, requirements, and safeguards planned or implemented within an AIS environment to ensure the integrity, availability, and confidentiality. This is one approach that will allow for integration and cooperative input from the CM, AIS Security Engineering, and QA management groups. Each of these product assurance functional support groups must understand and embrace common corporate product assurance objectives, synergize resources, and emerge as a partnership pursuant of corporate political strife dedicated to providing a harmonization of systems integrity, availability, and confidentiality.

The harmonization effort evolves as an enterprise-wide New Alliance Partnership Model (NAPM) in which:

QA provides an enhanced product assurance visibility by ensuring that the intended features and requirements, including but not limited to security, are present in the

delivered software. QA allows program management and the customer to follow the evolution of a capability from request through requirement and design, to a fielded product. This provides management with an enhanced capability as well as a forum, for identifying and minimizing misinterpretations and omissions which may lead to vulnerabilities in a delivered system. The formal specifications required by QA increase the chance that the desired capabilities will be developed. The formal documentation of corrective actions from reviews (of specifications, designs, etc.) lessens the chance that critical issues may go undetected.

- CM provides management with the assurance that changes to an existing AIS are performed in an identifiable and controlled environment and that these changes do not adversely affect the integrity or availability properties of secure products, systems, and services. CM provides additional security assurance levels in that all additions, deletions, or changes made to a system do not compromise its integrity, availability, or confidentiality. CM is achieved through proceduralization and unbiased verification ensuring that changes to an AIS and/or all supporting documentation are updated properly, concentrating on four components: identification, change control, status accounting, and auditing.
- AIS security provides additional controls and protection mechanisms based upon system specifications, confidentiality objectives, legislative requirements and mandates, or perceived levels of protection. AIS security primarily addresses the concerns associated with unauthorized access to, disclosure, modification, or destruction of sensitive or proprietary information, and denial of IT service. AIS security may be built into, or added onto, existing IT or developed IT products, systems, and services.
- Organizational management provides the empowerment and guidance for the economies of scale.

A seminal case study is presented as proof of concept for gaining security integrity assurance. It identifies the interdependencies and synergy that exist between the CM, AIS Security Engineering, and QA functional management activities. It describes how IT, as a principle change driver, is forcing the need for a QA, CM, and AIS security forum to evolve if the enterprise is to be successful in providing high-integrity systems.

The security control architecture (SCA) is the authorized mechanism used for baselining DPI system security architectures at the NASA JSC MOD and serves as the means for accrediting both operations and development environments. Such DPI system security architectures would include the Mission Control Center (MCC) as it exemplifies a life/mission critical system with both types of environments. Many challenges were encountered throughout the process of institutionalizing the SCA tool. For the intended purposes of this paper, focus has been placed on how AIS security features are input to the baseline security architecture, implemented, and tested and validated. Additionally, the process used for managing software and hardware change while maintaining the integrity and availability of life/mission critical systems was another very important point

of interest. Finally, a status for the NAPM implementation process has been provided. The process has not been an easy one, nor one without challenges. It is not yet complete; however, NAPM has proven to be an effective approach to managing the integrity and availability of high-integrity unclassified systems and may also be applied to classified systems.

MCC Support Request (SR) Process

The Security Engineering and Operations (SE&O) organization, the AIS security functional team member at NASA JSC MOD, is responsible for facilitating and maintaining all SCA activities for each MOD DPI at JSC. The MOD is responsible for planning, directing, managing, and implementing all mission operations activities including developing and operating all ground facilities. The support request (SR) is the authorizing document with initiates change within all NASA JSC MOD DPIs. SE&O is an integral member of this process from start to end. Both the MCC DPI computer security official (CSO) and the designated SE&O representative have the opportunity to review each MCC SR submitted by sustaining engineering, provide any applicable AIS security requirements as prescribed in the center security manual (JSCM 2410.11), review test scripts, and participate in the testing and verification of AIS security features.

The DPI CSO is tasked to review all SR's initiated in his/her operations center to input AIS security requirements. This is achieved by completing an AIS Security Checklist and attaching the checklist to the SR as a bonafide addendum set of requirements which are given full consideration by the responsible engineering support organizations. The AIS Security Checklist is a comprehensive form that was designed to communicate AIS Security requirements to all responsible hardware and software engineering organizations.

AIS Security Checklist Process

When NASA Facility Management requests an enhancement, removal, and/or otherwise change to the baseline configuration of the DPI, the SOC Configuration Management (CM) functional support team member is notified. This team member is responsible for maintaining the baseline configuration for all MOD Space Shuttle and Space Station support systems. The CM functional support team member is provided an approved SR to officially begin the process of implementing change the baseline configuration. An initial distribution of this acknowledgment is made, by the CM functional support team, to other key organizations including the AIS Security functional support group (SE&O). SE&O uses this opportunity to perform an impact analysis and provide AIS security requirements via the AIS Security Checklist (ASC) where needed.

The DPI CSO evaluates the SR for security impact and qualitatively determines and identifies the AIS security impact level to be either none, minor, and major. If significant minor or major DPI changes are identified from an AIS security standpoint, AIS security requirements, as per JSCM 2410.11, are then stated and delineated on the ASC. Upon completion of the DPI CSO evaluation, the ASC is then attached to an SR as an addendum set of AIS security requirements and becomes an integral component of the engineering requirements set. Additionally, the cognizant DPI CSO will indicate on the

ASC whether the new AIS security requirements will drive hardware and/or software support activity when an SR is approved for implementation. The cognizant DPI CSO will determine, based on the security impact level and how the baseline Security Control Architecture, AIS Security Procedures, and Disaster Recovery Plan are impacted. Finally, the cognizant DPI CSO may also request specific interest in being present for the implementation and testing phases of an SR and/or to be notified of SR close-out activity.

The SR (engineering requirements set) is then received by the DPI Operations Center's lead engineer for Rough Order Magnitude (ROM) costing analysis by the hardware and software engineering support teams. This effort is facilitated and coordinated by the lead engineer(s). It is important to recognize that this activity is one of several hinge pins which determine the success or failure of achieving a closed loop process. The communication process between the hardware and software engineering support teams and the AIS Security team member must be assured. Without buy-in from the key software and hardware support organizations, there is little expectation of finding an acceptable level of integrity assurance. When all ROMs for related cost are input into a roll-up total cost figure, then the subject SR is presented to the NASA DPI facility manager for approval and implementation.

The subject requirements are then disseminated to the responsible hardware and software support engineering management team for action. Once the engineering requirements set are communicated to, received by, and understood by the responsible hardware and software support engineering team members, internal task orders are generated to document the affected work group(s) and in general terms what work is to be accomplished.

Hardware and software support engineering work group(s) use internal task orders as input to develop more detail implementation instructions and test script procedures. Detail hardware and software implementation instructions are coordinated and formalized in design reviews. All documentation generated in support of an SR is evaluated and considered by all engineering disciplines during this period. During this feedback period, the AIS Security team member may identify a deficiency and notify the respective hardware and/or software engineering support group(s) of the correction or modification. When a final detail set of implementation instructions and test script procedures are refined to an acceptable state, the implementation and testing phases of an SR begins. The cognizant DPI CSO will be notified of this activity if he/she has expressed interest, on the ASC, to be present. Otherwise, the QA team member has third party responsibilities for witnessing all implementation and test verification activities. The QA team member is also tasked to notify the AIS Security organization of any unsuccessful implementation or test script procedures for closed loop purposes.

The process aforementioned is ideal and facilitates the integrated and cooperative input from the CM, AIS Security, and QA management groups. However, each of these key functional support groups must understand and embrace common corporate product assurance objectives, synergize resources, and emerge as a partnership pursuant of

corporate political strife dedicated to providing a harmonization of systems integrity, availability, and confidentiality. At a closer look, some real-world experiences with developing, producing, and maintaining high integrity IT systems may offer insight to the issues that undermine the effectiveness of corporate product assurance initiatives. One such example has been provided to understand the challenges of assuring the integrity of life/mission critical IT systems is the MCC at NASA JSC.

The ASC Experience

A modified ASC was first introduced to the SE&O sometime in mid 1992 as a working document. The ASC has not been well understood by the engineering community on the whole since that time. However, much has been learned through process improvement initiatives targeted at facilitating improved communications between the key functional support and hardware and software engineering organizations. These initiatives were orchestrated through management involvement.

In terms of what was not understood about the ASC process, given the number of participants, it seems that each respective team member in the ASC process had a perception of how it all worked. In mid 1994, the SE&O management team formed a process improvement team to determine how a closed-loop ASC process should function. After extensive research, it was determined that several process disconnects and gaps existed causing serious uncertainty and doubt as to whether AIS Security requirements were actually considered in the SR process. As the process improvement team identified each functional support player in the ASC system, an open-loop process unfurled. Interviews were arranged soon thereafter with each process management team member to gain a more accurate perspective of the ASC system. Additional opportunity was introduced with every interview the AIS Security team facilitated.

Configuration Management

The interview process began with the CM team member who is chartered to maintain the baseline configuration and associated supporting documentation. The process improvement team learned that CMs scope and sphere of influence could not provide at any level of certainty that the ASC was in fact being considered and treated as a bonafide engineering requirements set. Further, CM management was unable to produce any evidence that AIS Security countermeasures had been implemented and/or tested from a documentation standpoint. Essentially, the CM functional support team was managing a repository of documentation, authorization paper trail, and engineering drawings which it received as input and used it to maintain the associated baseline hardware and software configuration. During these discussions, the process improvement team learned that the CM team operated with certain seeded beliefs that the engineering support organizations, QA organizations, and SE&O organization had accountability for evidencing the implementation and testing of AIS security countermeasures.

In part, the CM management team is correct; however, can not ignore how its involvement with the original assignment of an SR. The CM group does not authorize any system change without sufficient documentation from all the engineering support

organizations and QA evidencing the implementation and testing of each engineering requirement as stated on an SR. The CM team maintains a standing rule of notifying the SE&O team whenever they have expressed interest on the ASC to close-out AIS security requirements stated on an SR. However, such close-out requests are few in number and the majority of SR's that have AIS security requirements have had no mechanism for assuring closure by the associated engineering support groups. The CM functional support organization is a pivotal control point in the ASC system. Without the functional support from such groups like the SE&O, QA, and engineering support organizations, the ASC system will not be fully effective until closure mechanisms are established between these groups. Otherwise, it becomes an increasingly more difficult task for a cognizant DPI CSO to assure that AIS security countermeasures identified in a respective DPI SCA are in fact implemented, tested successfully, and functioning properly.

Hardware Engineering

In other interviews with the hardware engineering support group, the SE&O team received another interesting data point. The SE&O organization realized that in fact it's own organization was responsible for a critical system disconnect. The SE&O organization had not been effective in providing an AIS security feedback mechanism for responding to implementation and test script procedures reviews. The SE&O made several attempts to close this loop; however, due to a variety of internal political issues being driven by the changing environments of the time were unsuccessful. This disconnect could have been, in part, minimized through more management training. As indicated on the ASC, when the cognizant DPI CSO had identified hardware security impact and testing requirements, the cognizant hardware engineer responded by sending an implementation and test script package for AIS Security technical evaluation and feedback. The internal hardware support team documentation was also determined to require several minor decision point modifications for facilitating a more fluid feedback process.

Software Engineering

Interviews with the software engineering support group were equally productive and valuable. In light of how NASA, JSC, and specifically MOD operational systems have undergone major IT system reconfigurations, the SE&O organization considers the software engineering type SR activity as the area which presented the highest level of uncertainty from an AIS security perspective. Prior to these interviews, the SE&O had very little insight, due to political and changing environments, for what software engineering support personnel did with the ASC. It was soon determined that there was no existing feedback communication means between the SE&O and the software engineering management team. It was like the SE&O had been sending three years worth of AIS Security resource into a void. The team learned that the ASC had not been well understood by the software support group since it's implementation. This was one of the most significant gaps identified by the process improvement team during its analysis.

The SE&O support group developed the ASC and introduced it to the SR process as the official process for establishing AIS security engineering requirements input. It was originally intended to serve as a bonafide addendum and part of the entire SR engineering requirements set. Well, what was learned did not quite meet this intent. Upon further discussions, the problems continue to unravel.

These discussions identified a real need to establish formal lines of communication with key area support management. Through this dialogue and mutual understanding, the software engineering team formed an internal process analysis team to gain a better understanding for how AIS security requirements were responded to internally from a management standpoint. This effort was also chartered determine how AIS security requirements were articulated, implemented, tested, and documented. In other words, the SE&O team was interested in a documentation trail and very interested as to how the SR engineering requirements set was being communicated downward for implementation. Specifically, how were SR high-level requirements translated and communicated internally to document the affected work group(s) and in general terms articulate what work was to be accomplished. In effect, the teams reached an impasse and continue to work the issue.

The SE&O organization had been even less effective in providing an AIS security feedback mechanism for responding to implementation and test script procedures reviews to the software engineering. As with the hardware engineering group, the SE&O made several attempts to close the gap with the software management team. However, once again the business of providing high integrity systems was clouded with a variety of internal political issues being driven by the changing environments. In the case of the software engineering support team, the SE&O organization had not received a single implementation and test script procedure for AIS security requirements review and evaluation during the past three years. Ideally, when a cognizant DPI CSO indicates Software AIS security impact and testing requirements on a ASC, the cognizant software engineer should respond by sending an implementation and test script package for a technical review and evaluation. And much like the hardware engineering support team the effects of this gap may have been minimize with more AIS Security awareness training. This experience was uncomfortable although a beneficial realization.

Quality Assurance

Another key player in the NAPM approach for assuring security integrity is the QA functional support organization. The QA support team is a pivotal control point in the ASC system. Discussions with the SE&O process improvement team were no less insightful. The team learned that QA personnel were very knowledgeable with the execution of SR engineering requirements sets and associated implementation and test script procedures. However, they did not recognize the ASC as a bonafide addendum to the engineering requirements set and failed to understand its significance. The QA support team, specifically the software QA function, was not in the loop to know the significance of the ASC. How had AIS security requirements been articulated, implemented, tested, and documented up to this point? AIS security requirements issues

had not been included in any of the established QA life cycle event checklist documents. What had the engineering support leads been providing the CM support function as documented evidence that all SR AIS security requirements had in fact been satisfied? Further, what collected evidence had been used to assure the security posture of a given DPI up to this point? The process improvement team had identified another gap in the ASC system and this one pertained to closure, the documented evidence of successfully implemented and fully tested AIS security countermeasures. This was of major significance, in that SE&O support team function had struggled for years to find that closure mechanism which could evidence the implementation and testing of AIS security countermeasures. Without such evidence of closure, the SCA approach for accrediting DPIs could also be weakened.

NAPM In Practice

At the outset of this initiative, there were serious uncertainties and doubts as to whether the AIS Security requirements set had actually been considered in the SR process. The SE&O team's attempt to hone in on a documentation trail had evidenced an open-loop communication process. Discussions with the QA team and others had validated a breakdown in communications between the CM, AIS Security Engineering, Hardware and Software Engineering, and QA as to the intent of the ASC. By gaining concurrence from the QA team that in fact the AIS security engineering requirements set was not being recognized as a bonafide addendum to the SR process, it was clear that the NAPM approach offered a qualified solution for improvement.

The NAPM approach for providing integrity assurance to mission and/or life critical systems presented significant opportunity. The NAPM purports to gain an understanding of the interdependencies and complimentary pursuits that exist between the CM, AIS Security and QA, organizational support functions. To these ends the SE&O support team applied the NAPM alternative to identify the synergy for realizing new economies throughout the hardware and software system life-cycle through continuous improvement techniques.

The SE&O process improvement team promulgated several ASC system gaps which share common themes, specifically in training and communication. Based on the collected input from the organizational support functions it was determined that AIS security requirements were not being communicated downward for implementation by the affected work group(s).

Prior to the introduction of the ASC system, CM management was unable to point to any evidence that AIS Security countermeasures had been implemented and/or tested from a documentation standpoint. The CM organization, SE&O organization, QA organizations, and engineering support organizations have a collaborative level of accountability for evidencing the implementation and testing of AIS security countermeasures. Based on this collective input, the CM support team uses it to maintain each DPI's hardware and software baseline configuration. Today, the CM group does not authorize any system change without this level of collaborative and collective documentation from the

participating functional support team who evidences the successful implementation and testing of each engineering requirement stated on an SR. The CM team continues to maintain the standing rule of notifying the SE&O team whenever close-out has been expressed on the ASC. This feedback mechanism has proven to be very effective for bringing closure to stated AIS security requirements. This kind of closure increases the confidence level in which a DPI CSO assures that AIS security countermeasures have been successfully implemented, tested, and functioning properly.

The SE&O organization worked intently with the hardware engineering support management team to refine the AIS security feedback process. The SE&O provided additional ASC training so as to detail the instructions for providing AIS security feedback during implementation and test script procedures reviews. The SE&O and hardware engineering management teams had successfully closed the open-loop through improved understandings of how to use the ASC and with minor modifications to the internal hardware engineering documentation. The functional relationship between these two team members has shown marked improvement in terms of effectiveness and cooperation. The ASC has become an excellent communication tool between these support functions regardless of political and changing environments of the time. After all improvement measures had been implemented, AIS Security technical evaluations of implementation and test script reviews began to feedback in unprecedented numbers. It should be noted that this process step continues to improve and refine itself through the increased communication and cooperation between the SE&O and hardware engineering support organizations.

The SE&O team has continued to build on its success of establishing communication lines to software engineering management team so as to refine the AIS security feedback process. The SE&O organization continues to facilitate meetings in support of this end. A considerable amount of progress has been accomplished through this effort. And much like the hardware engineering support team, additional AIS Security awareness and ASC training was provided so as to detail the instructions for providing AIS security feedback during implementation and test script procedures reviews. This experience was uncomfortable although a beneficial realization. The process improvement team continues to identify opportunities and solutions with the Software Engineering support team.

After several productive meetings, the software QA management team agreed that the AIS security requirement set identified on the ASE should be viewed as part of the SR requirements set. Up to this point, AIS security requirements issues had not been included in any of the established software QA life cycle event checklist documents. The QA support team agreed to modify their checklists to include additional QA steps to assure that AIS security requirements are responded to in future events. This type of corroborating evidence and other collected documentation have significantly contributed to assuring the security posture of a given DPI. The ASC system is now the official communication tool for bringing change to the software QA support function. The ASC has also been instrumental in providing evidence of closure to the CM support team that

AIS security countermeasures have been successfully implemented and fully tested. If however, the implementation and testing of the prescribed countermeasures are unsuccessful, the QA team is still tasked to notify the AIS security organization for resolution.

Summary

Change is not easy. Change has not been easy. Change will not be easy. In this case study, the members of each respective management support team have championed the process improvement initiatives and the corrective actions taken thus far. It is important to emphasize that employee empowerment of this type must be supported by top management because security integrity engineering and the implementation of an integrated product assurance and secure information technology development process such as a control architecture is a proactive management decision.

As management continues to learn more about the interdependencies and common pursuits that exist between the Configuration Management, Automated Information System Security Engineering, and Quality Assurance organizational functions, it will realize additional opportunity for economies through continuous process improvement techniques.

Information technology has been and will continue to be a major change driver that establishes a need for a functional organizational support forum dedicated to delivering high-integrity products and services. Each of the product assurance functional support organizations must understand and embrace common corporate product assurance objectives, synergize resources, and emerge as a partnership independently pursuant of corporate political strife and dedicated to providing a harmonization of systems integrity, availability, and confidentiality.

The New Alliance Partnership Model is a viable solution that has been put to the test and proven in a highly dynamic operational environment of ever-changing distributed processing technologies. The NAPM supports the integration process and requires that direct lines of communication be bridged between key functional support organizations so as to input and feedback closure information.

The Automated Information System Security Checklist is an excellent tool for assuring AIS security feedback to key hardware and software engineering support functions when reviewing SR implementation and test script procedures for AIS security impacts. The ASC has become an excellent communication tool between these support functions regardless of political and changing environments of the time. The ASC functions as a key communication tool for facilitating the initiation, implementation, testing, and documentation of any AIS security requirements set. In this case, the CM team is an integral key player in the closure of AIS security requirements which are stated on an SR. The ASC system is not fully effective until closure mechanisms are established between the functional support groups like the CM, SE&O, QA, and engineering support organizations. The security control architecture methodology used for baselining and

maintaining an accreditable data processing installation is highly dependent on the delivery of documented evidence supporting the successful implementation and testing of AIS security countermeasures.

In conclusion, business enterprises must always be assured of a capability to maintain an AIS Security Control Architecture baseline to maintain stability for growth and strategic competitiveness.

AN UNUSUAL B3-COMPLIANT DISCRETIONARY ACCESS CONTROL POLICY¹

Jeremy Epstein, Gary Grossman, and Albert Donaldson
Cordant, Inc.²

ABSTRACT

There are many possible identity-based discretionary access control (DAC) policies. This paper describes an unusual DAC policy: rather than associating access control information with the objects (e.g., files) in the system, access control decisions are based on pattern matching against a centralized database. This policy has certain advantages and disadvantages compared to more common (e.g., UNIX) access control policies, which are explained. While not an original goal of the design, the policy meets the TCSEC B3 functional criteria for DAC.

1. Introduction

There are many possible discretionary access control (DAC) policies. Among the more common are permission bits³ (e.g., UNIX, older VMS versions) and access control lists (e.g., Multics, newer versions of DEC VMS and Novell NetWare). In each of these, access control information is associated with the objects in the system, typically files. In most cases, every object has its own access control information, which is stored with the object⁴.

Our system architecture imposed several constraints that made such an approach impractical. Because our system is based on a personal computer DOS (e.g., MS-DOS) file system, there is no way to effectively store access control information with the file, because there is no empty space in the File Allocation Table (FAT) entry. For compatibility reasons, we did not want to modify the file system structure. We also wanted an easy way to cause changes made by an administrator at a central server to apply to all workstations in a network. Because our workstations interact with the server in an asymmetrical client-server manner, we could not rely on notifying workstations of access control changes, but rather need to allow them to download access control information at appropriate points.

For the past several years, Cordant has developed and marketed a product line under the brand name Assure⁵ which meets each of these objectives. For product line compatibility, we wanted to use that product as a starting point. However, there were several problems with that product. While it has all the necessary features to meet the *Trusted Computer System Evaluation Criteria* [TCSEC] Class C2 criteria, it can not meet the Class C2 assurance criteria because it lacks a Trusted Computing Base (TCB). Secondly, and more importantly from the perspective of this paper, the Assure policy is not fully defined:

¹Copyright © 1995 Cordant, Inc.

²11400 Commerce Park Drive, Reston Virginia. Mr. Epstein: 703-758-7367; jepstein@cordant.com. Mr. Grossman: 703-758-7363; ggross@cordant.com. Mr. Donaldson: 703-758-7000 x7227; al@escom.com.

³Permission bits are arguably an access control list with a fixed number of entries and specific uses for each entry.

⁴Sometimes, as in the case of certain UNIX systems based on SecureWare or AT&T System V MLS technology, the complete access control information is stored in a database, with each object only containing a tag to indicate the database entry to be used. Nonetheless, each object contains some level of access control information.

⁵Assure is a registered trademark of Cordant, Inc.

there are subtle cases where the results of an access control decision are indeterminate, given a particular configuration⁶.

The remainder of this paper describes our file system object⁷ access control policy, which is implemented in a forthcoming product, Assure EC^{TM8}. The paper is organized as follows: Section 2 summarizes the network architecture of which the Assure EC workstation is a part. Section 3 describes the access control policy. Section 4 describes some of the interesting aspects of the policy, including how it meets the B3 criteria. Section 5 concludes the paper.

2. Network and Component Architectures

The Assure EC workstation is part of Novell's Trusted NetWare network architecture. Trusted NetWare is being evaluated against the *Trusted Network Interpretation* [TNI] as a Class C2 network. There are two types of active components in Trusted NetWare: workstations and servers. Both types of active components have NTCB partitions. Workstations must be at least "I" components, and servers must be at least "IAD" components. Workstations and servers communicate in a client-server architecture, where clients make requests of servers, but servers never send requests to clients. Servers provide facilities that can be used by both trusted and untrusted software running on workstations. These facilities include storage of files, configuration information, and audit data.

The initial evaluation includes one server (Novell NetWare) and one workstation (Cordant Assure EC), both running on generic IBM PC computers. The Assure EC workstation is an "ID" component. The Cordant workstation relies on the NetWare server for storage of TCB configuration data and audit data.

Further information on the network architecture can be found in [NetArch]; a description of the component architectures can be found in [CompArch].

3. The DAC Policy

The DAC policy enforced by Cordant's Assure EC product is based on pattern matching of file names. Administrators define file path name patterns, and the file rights associated with those patterns using a menu driven application. When a user logs in, the workstation TCB looks up the patterns associated with the user and all groups of which the user is a member. The combination of user and group patterns is then used for making access control decisions for the duration of the login session.

3.1. DOS File and Path Naming

DOS provides a hierarchical file system. File names consist of a base name, consisting of one to eight characters, a period, and an optional extension, consisting of one to three characters. The period that separates the base name and extension can be omitted if there is no extension. The base name and extension can include letters, numbers, and a variety of special characters, and are case insensitive. Table 1 shows valid and invalid file names.

Table 1: Valid and Invalid File Names

File Name	Explanation
foo.bar	Valid; same as FOO.bar, FoO.Bar, etc.

⁶Of course they are not random, but the results cannot be determined by a user or administrator, since they rely on internal ordering of data structures, which cannot be determined using human interfaces.

⁷The forthcoming product has other types of objects besides file system objects, each of which has its own DAC policy. However, those policies are relatively uninteresting, and are not explained further in this paper.

⁸Assure EC is a trademark of Cordant, Inc.

abcdefghijkl.	Invalid; too many characters in base name ⁹
abc.def.ghi	Invalid; only one extension allowed
abc_def.ghi	Valid; underscore can be used as a separator
abc?def	Invalid; ? is not a valid character in a file name
abc.*	Invalid; * is not a valid character in a file name

A full path name consists of a drive letter (which identifies a particular logical disk drive), followed by a colon, a backslash, and a series of file names separated by backslashes. The special file names "." (dot) and ".." (dot-dot) which represent the current directory and parent directory, respectively, are not valid in full path names. Table 2 shows several valid and invalid full path names.

Table 2: Valid and Invalid Full Path Names

Path Name	Explanation
\foo.bar	Invalid; doesn't have drive letter
c:\foo.bar	Valid
d:\foo\bar	Valid
c:\\foo\bar	Invalid; can't have multiple sequential backslashes
c:\foo\.\bar	Invalid; can't have dot or dot-dot in path name
c:\foo\bar\	Invalid; can't have trailing backslash

3.2. File Name Patterns

File name patterns are not DOS file names. Rather, they are used by the Assure EC product for assigning rights. A pattern is defined as a DOS full path name, with the following changes:

- The drive letter may be replaced by a question mark.
- The base name and/or extension in the final component of the path (i.e., after the last backslash) may be replaced by a string that terminates with a "*" (star).

Table 3 shows some valid and invalid file name patterns.

Table 3: Valid and Invalid Patterns

Pattern	Explanation
c:\foo\bar	Valid
c:\foo\bar.a*	Valid
?:\foo\abc*.bar	Valid
:\foo\bar	Invalid; "" cannot appear as drive letter
?:\foo\bar.??	Invalid; "?" cannot appear except as drive letter
?:\foo\abc*def.*	Invalid; the "*" (if used) must terminate the base name

Note that certain patterns are invalid as DOS file names. For example, abc.* is a valid pattern, but not a valid file name¹⁰.

3.3. File and Directory Rights

Associated with each pattern in the access control database can be zero or more file and directory rights. Table 4 describes the meaning of the right if associated with a file or directory.

⁹Long names may be truncated by applications, but DOS itself will refuse a name of this form.

¹⁰A name such as abc.* can be used as a wildcard to a command (e.g., "copy abc.*"), but the star is not part of the file name.

Table 4: File and Directory Rights

Right	Meaning for Files	Meaning for Directories
Read	File can be read	Unused
Write	File can be written	Unused
Scan	File name can be seen	Directory name can be seen
Delete	File can be deleted	Directory can be deleted
Rename	File can be renamed	Directory can be renamed
Create	File can be created	Directory can be created

Note that because patterns are not associated with objects, it is possible to express the ability to create objects before they exist. That is, it is possible to say that a user can create \FOO\BAR without making any statement about the user's rights to \FOO, or about that user's ability to create \FOO\XYZ. This is different from other systems, such as UNIX, where the ability to create a file (or subdirectory) in a directory means that any name can be used, so long as it does not already exist.

3.4. Path Records

A path record consists of three parts: a pattern, a (possibly empty) set of file and directory rights, and (optionally) an encryption key. If the encryption key is present, all files and directories that match the pattern are DES [FIPS46] encrypted in electronic codebook [FIPS81] mode using the key provided. File encryption is invisible to the application software: files are automatically decrypted as they are read, and encrypted as they are written. If there is no encryption key, then the files are not encrypted.

3.5. Pattern Ordering

Users are associated with zero or more groups. Path records can be associated with users, with groups, or both. When a user logs in, the TCB consults a database to find out the groups that user belongs to. It then finds the path records associated with the user and all groups of which the user is a member. The resulting list is sorted as follows:

- User-specific path records are divided into those with patterns containing wildcards ("?" and/or "*") and those not containing wildcards. Each list is then sorted by length from longest to shortest pattern. In the case of the wildcarded list, the characters "*" and "?" sort after all other characters in the lexicographic order. This results in each list being sorted in order of decreasing specificity.
- Group-specific path records containing identical patterns are merged, with file rights ORed together. The resulting list is then sorted identically to user-specific path records.

The four lists are then concatenated in the order user non-wildcarded, user wildcarded, group non-wildcarded, group wildcarded. The result, known as the *consolidated access list* (CAL), is the access rights for the session.

Table 5 shows some sample user path records and group path records.

Table 5: User and Group Path Records

User/ Group	Pattern	Rights
Alice	C:\DOS	Scan,Read, Write
Alice	C:\DOS*.EXE	Scan,Read,Rename
Alice	?:\DOS\SORT.EXE	Scan,Read,Write,Delete
Bob	?:\FOO	Scan,Read,Write,Delete
Bob	?:\DOS	Scan,Read,Rename
Mgmt	?*.*	Scan,Read

Mgmt	?:\DOS	Scan,Delete,Create
All	?:\DOS	Scan,Read
All	?:\WINDOWS	Scan,Read
All	?:\WINDOWS*.INI	Scan,Read,Write,Create

Given that *Alice* is a member of groups *Mgmt* and *All*, and *Bob* is a member of group *All*, Table 6 shows the CAL for each user.

Table 6: Consolidated Access Lists

User	Pattern	Rights	Origin	Comments
Alice	C:\DOS	Scan,Read,Write	Alice	User-specific, no wildcard
	?:\DOS\SORT.EXE	Scan,Read,Write,Delete	Alice	User-specific, wildcarded
	C:\DOS*.EXE	Scan,Read,Rename	Alice	Shorter than previous pattern
	?:\WINDOWS*.INI	Scan,Read,Write,Create	All	Longest group pattern
	?:\WINDOWS	Scan,Read	All	Next longest group pattern
	?:\DOS	Scan, Read, Create, Delete	Mgmt + All	Identical pattern in Mgmt and All, so rights ORed
	?*.*	Scan, Read	Mgmt	"*" sorts after all other characters
Bob	?:\DOS	Scan, Read,Rename	Bob	User-specific, wildcarded
	?:\FOO	Scan, Read, Write, Delete	Bob	FOO sorts after DOS
	?:\WINDOWS*.INI	Scan,Read,Write,Create	All	Longest group pattern
	?:\WINDOWS	Scan,Read	All	Next longest group pattern
	?:\DOS	Scan, Read	All	Bob is not in Mgmt, so only gets rights from All

Note that the C:\DOS and ?:\DOS patterns are not merged for *Alice*, because they are not identical.

3.6. Run-Time Pattern Matching

To access files or directories stored on the local disk¹¹, application software creates the full path name being accessed (i.e., the path name is canonicalized). The full path name is then transmitted to the TCB, which compares the full path name to the list of patterns to determine the access rights available to the user.

The pattern matching algorithm compares the full path name to each of the records in the CAL. If the full path name is a superstring of the pattern in the path record, or matches an expansion of the wildcards in the pattern (where "?" matches a single character drive name, and "*" matches any sequence of characters in the file name), then it is considered to have matched, and the rights associated with the path record are the rights available to the user. Only the first path record to match is used; any subsequent rights (greater or lesser) are ignored. If the end of the CAL is reached without any match, then the user has no access to the requested path name.

Table 7 shows some patterns matched against the CALs shown in table 6, and the resulting rights available to the user.

Table 7: Pattern Matching

User	Path Requested	Pattern Matched	Rights Granted
Alice	C:\DOS\SORT.EXE	C:\DOS	Scan,Read,Write

¹¹ Application software can also access files and directories stored in NetWare servers. That access control policy is under the control of the NetWare server, and is not further discussed here.

Alice	C:\WP\WP.EXE	?:*.*	Scan,Read
Alice	D:\DOS\PRINT.EXE	?:\DOS	Scan,Read,Create,Delete
Bob	C:\WINDOWS\WIN.INI	?:\WINDOWS*.INI	Scan,Read,Write,Create
Bob	C:\WINDOWS\FOO.BAR	?:\WINDOWS	Scan,Read
Bob	D:\FOO\HELLO.TXT	?:\FOO	Scan,Read,Write,Delete
Bob	D:\WP\WP.EXE	None	None

Note that *Alice's* rights are constrained by the pattern C:\DOS. Although she has Rename rights to C:\DOS*.EXE, Delete rights to ?:\DOS\SORT.EXE, and Create rights to ?:\DOS (by virtue of her group membership in *Mgmt*), she can never gain any of these when accessing files in C:\DOS, which is a non-wildcarded pattern and therefore matches first. Changing the pattern C:\DOS to ?:\DOS or C:\DOS*.* would make it wildcarded, and therefore yield different results.

4. Policy Discussion

In this section we discuss several interesting (and sometimes surprising) aspects of the DAC policy.

4.1. Unusual Ordering

Note that the combination of CAL ordering and the pattern matching capability has several somewhat surprising results:

- The strings \ and *.* are not the same, although they will match the same values. However, \ is a non-wildcarded pattern, and hence will appear in the CAL *before* *.*, which is a wildcarded pattern (presuming that both apply to the same user, or both apply to groups).
- The policy denies all rights not explicitly granted. This is easy to reverse, by adding a path record including the pattern ?:*.* to a group that the desired user(s) are member(s) of. Note that adding this path record for a user would cause any path records associated with groups to be ignored, since user patterns are matched before group patterns.

4.2. Encryption Facilities

Any path record can include an encryption key. If present, files matching that path record are automatically decrypted when read and encrypted when written. This allows applications to run unchanged.

Because encryption keys are associated with patterns, and not with particular files, it is possible to configure the product so that some people see the encrypted file contents, while others have the files automatically decrypted when reading. Figure 1 shows some examples of this feature. If *Bob* has a pattern for C:\DATA that includes an encryption key, then access by *Bob* to anything in C:\DATA will automatically be encrypted and decrypted as necessary. An *Operator* could be given the ability to back up data without decrypting what is being backed up by giving them the Read rights to C:\DATA, but without an encryption key. Then *Operator* would only have access to the encrypted version of the file.

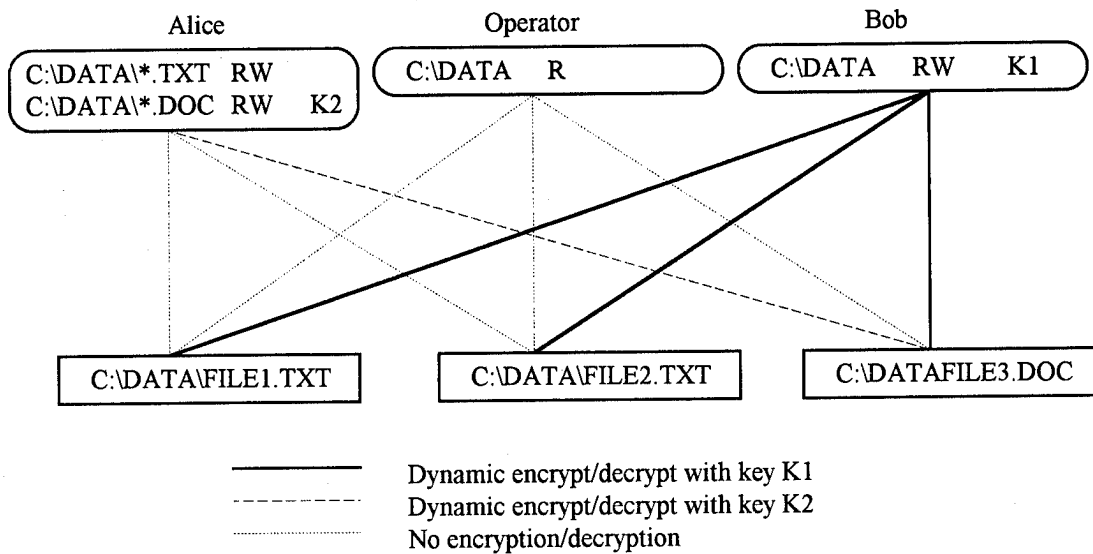


Figure 1: Transparent File Encryption

A problem with this scheme occurs if *Alice* has rights for the pattern `C:\DATA*.TXT` without an associated encryption key. Then *Alice* will see any existing files that match the pattern without being automatically decrypted, and any files she creates will not be encrypted. Similarly, if *Alice* has rights for the pattern `C:\DATA*.DOC` using an encryption key *K2*, rather than key *K1* used by *Bob* for the pattern `C:\DATA`, they will be unable to share files, since files that *Alice* creates will appear to be garbage to *Bob*, and files that *Bob* creates will appear as garbage to *Alice*.

Another problem occurs if two groups are set up with different keys for the same pattern. If *Mgmt* has been rights for the pattern `C:\DATA` using encryption key *K3*, and *All* has rights for the pattern `C:\DATA` using encryption key *K4*, then it is non-deterministic whether key *K3* or *K4* will be used for users who are members of both *Mgmt* and *All*.

These are both inherent aspects of the design. Administrators are cautioned not to use multiple encryption keys for potentially overlapping paths, lest they encounter the problems described here.

4.3. The B3 DAC Criteria

Products evaluated as TNI "D" components can be rated as C2+, rather than C2, if they meet the TCSEC Class B3 DAC feature requirements. If a product receives a C2+ rating, it can be integrated with other components rated B3 or A1 to produce a system which as a whole is rated B3 or A1. However, if the product only receives a C2 rating, then combining it with other components rated B3 or A1 will yield at most a system rated B2. Hence, it is useful from an integration standpoint to produce C2+ components, rather than C2 components.

The TCSEC B3 DAC criteria require the ability to allow or deny access to users or groups. Section 3.3.1.1 of the TCSEC (and TNI) state:

These access controls shall be capable of specifying, for each named object, a list of named individuals and a list of groups of named individuals with their respective modes of access to that object. Furthermore, for each such named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is given.

Because the Assure EC DAC policy allows specifying null rights, and uses a first match capability, it can be used to meet this requirement. The ability to *allow* rights on a user or group basis is clearly present. Showing the ability to *deny* rights to a user or group is slightly more complex. For example, to deny *Alice*

all rights to a file, the administrator can assign *Alice* a pattern with no rights. Because user patterns match before group patterns, and non-wildcarded patterns before wildcarded patterns, the explicit pattern will override any other rights granted to *Alice*. To deny group *Mgmt* all rights to a file, the administrator can assign *Mgmt* a pattern with no rights. However, if any user has been given rights to the file, or to a pattern that matches the file, then they will still gain access to the file by virtue of their individual assignment, regardless of the group membership. The wording of the TCSEC and TNI is not clear, but precedent indicates that there is no requirement that group rights override individual user rights.

4.4. ACL or Capability?

At several points in our design, we debated whether the path records are capabilities or access control lists (ACLs). Using a matrix of subjects and objects (as in [HRU]), where subjects are listed down the side and objects across the top, systems have capabilities if they store data by row (i.e., with the subject), and ACLs if they store data by column (i.e., with the object).

However, our path records do not meet the conventional view of a capability. In traditional capability systems, untrusted software is permitted to hold the capability, which is cryptographically sealed. Users may pass the capability, and thereby pass access rights. Additionally, capabilities are valid until explicitly revoked by an administrator.

In the Assure EC system, users do not have access to path records, and cannot pass their rights from one user to another. While path records are relatively permanent, the calculation of the CAL is performed on a per-session basis, and access to a path cannot be revoked until the end of the session. Thus, we believe that our path records are neither fish nor fowl: they have some of the characteristics of capabilities, but also have certain aspects of ACLs.

4.5. Central Database Usage

In a traditional stand-alone computer, all security configuration databases are kept in that computer. In a distributed system, it is desirable to have access to configuration databases from anywhere in the system. This is particularly true in a network with many personal computers. It would be undesirable for an administrator to have to visit each workstation in order to change the access rights available to a user. In a peer-to-peer network, changes could be propagated from one client to another. However, this technique is not feasible in a client-server architecture.

The Assure EC product stores its administrative information (e.g., path records) in a central repository, known as the NetWare Directory Services (NDS) Directory Information Base (DIB). Each workstation downloads the relevant information whenever a user logs in. Thus, changes to path records normally occur when a user logs in¹². That is, if a change occurs in the central database during a session, users who are already logged in to Assure EC workstations are not affected. Rather, the changes take effect the next time a user logs in, when the databases are downloaded from the DIB.

Because path records are not tied to a particular workstation, and the path records are stored centrally, all workstations in an administrative domain must be configured identically. That is, if user *Alice* is given rights to \DOS, then she will have those rights to \DOS on any workstation for which she is an authorized user. If administrators maintain consistency of configuration (which is a good idea in large networks), then the common paths will not cause problems either of granting too much or too little access. However, if each workstation in an organization has a different file organization, administrators will have a difficult time setting up the desired patterns and rights.

¹²If the workstation NTCB is unable to contact a server to download new information, it uses the cached information from the most recent download. This allows workstations to be used when disconnected from a network.

4.6. Administrative Control of Policy

In the Assure EC product, only administrators can create or modify the access control information. There are two reasons for this restriction.

First, because access control information is kept in the NDS DIB (see above), it would require that any user who has the ability to modify access control information would need to have access to modify the DIB. Since all paths are kept in a single storage location in the DIB, any user who can write that data can modify all path rights. Thus, for *Alice* to give *Bob* access rights, she would have to modify *Bob*'s information in the DIB, and could give *Bob* rights to anything she wanted.

Second, even if there were an effective way to control access to the information stored in the DIB, users can only see the workstation they are using, and not all of the workstations that the access controls apply to. That is, if *Alice* could give *Bob* rights to directory \MEMOS, then *Bob* would gain access to \MEMOS on all workstations he can use, not just to the copy of \MEMOS on the local workstation. Since *Alice* may be unaware of what is in \MEMOS on workstations other than the one she is using, it would be undesirable for her to give away such access. This is unlike more traditional systems, where access controls only apply to a single computer.

4.7. Looking Down the Tree

Suppose user *Alice* has Read and Scan rights to directory \DOS, and no rights to anything else in the system. If *Alice* attempts to get a list of files in \, she will see \DOS, and nothing else. This occurs because the scan operator is defined as requiring access to the objects whose names are returned (i.e., \DOS), and not to the directory being scanned (i.e., \). If *Alice* attempts to change her current directory to \DOS, the operation will be successful. This is an entirely consistent view to the user.

However, if *Alice* attempts to change to her current directory to \, that request will fail, because *Alice* has no rights to \, and changing directories requires that the user have at least one right. Thus, *Alice* will have a surprising result: she can see \DOS, and can change to \DOS, but cannot change to \.

This is not an unknown feature. Multics offers a similar feature, where users may be able to go directly to a destination directory without having access to intermediate directories. Novell NetWare has a similar feature, but handles it differently: a user implicitly has Scan rights to all directories along the path to every file that he/she has any rights to. That is, the presence of Read rights to file \A\B\C\D implicitly gives the user Scan rights to \A, \B, and \C.

5. Conclusions

There are many possible DAC policies. We have explained one such policy that offers some unusual features:

- The ability to coherently and efficiently manage access rights associated with a set of workstations from a central point.
- The ability to meet the B3 DAC criteria, and thereby participate in a B3 or A1 network.

6. Acknowledgments

The work described in this paper reflects many contributions. Marv Schaefer helped us see many of the weaknesses and undefined areas in the DAC policy enforced by our current product. Jon Dellaria, Dave Bixler, Mike Newman, and Bruce McKinstry all helped us to understand the subtleties of the existing security policy, and advised us on potential flaws. Anonymous reviewer #51 also had many useful comments that helped improve this paper.

7. References

- [CompArch] "Component Architectures for Trusted NetWare", Jeremy Epstein, Gary Grossman, and Roger Schell, in *Proceedings of the 18th National Information Systems Security Conference*, Baltimore MD, October 1995.
- [FIPS46] *Data Encryption Standard (DES)*, Federal Information Processing Standards Publication (FIPS PUB) 46-1, 1988.
- [FIPS81] *DES Modes of Operation*, Federal Information Processing Standards Publication (FIPS PUB) 81.
- [HRU] "Protection in Operating Systems", Michael Harrison, Walter Ruzzo, and Jeffrey Ullman, *Communications of the ACM*, August 1976, Volume 19 No 8.
- [NetArch] "An Open Trusted Enterprise Network Architecture", Gary Grossman, Jeremy Epstein, and Roger Schell, in *Proceedings of the 18th National Information Systems Security Conference*, Baltimore MD, October 1995.
- [TCSEC] *Department of Defense Trusted Computer System Evaluation Criteria*, National Computer Security Center, December 1985.
- [TNI] *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, Version 1*, National Computer Security Center, July, 1987.

GENSER MESSAGE MULTI-LEVEL SECURE (MLS) CLASSIFICATIONS AND CATEGORIES

Speaker

Mary Lou Hoffert

Authors

NCPII Development Team

Organizational Affiliation

NCTAMS LANT - Software Technology Department
NCTS Washington - Systems Development Directorate

Phone Numbers

Voice - 804-444-4638

DSN - 564-4638

FAX - 804-444-1427

Internet Address

Joelle_Griffith_at_NCTAMSLANT@NCTCGW.NAVY.MIL

Point of Contact

Joelle Griffith

US Government Program Sponsor

Naval Computer and Telecommunications Command

ABSTRACT

This paper proposes a labeling schema for a Multi-Level Secure (MLS) system processing and routing General Service (GENSER) messages for the DoD, Allied and NATO communities. It assumes an automated system integrating a dedicated application and trusted Commercial Off-The-Shelf (COTS) software products evaluated by the National Security Agency (NSA) at the B1 level. It further assumes that the application described would pass Defense Message System (DMS) security testing. It has attempted to provide a generic schema that could be employed DoD wide to facilitate intercommunication of MLS systems with no need for intermediary mapping translations. The discussion concentrates on GENSER message processing, but is expandable to DSSCS systems.

**GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)**

KEYWORDS

Allied	MLS
AUTODIN	multi-level secure
B1	NATO
category	NSA
classifications	object
CMW	Orange Book
Compartmented Mode Workstation	RESTRICTED
CONFIDENTIAL	SECRET
default category	security levels
Defense Special Security Communications System	sensitivity label
DoD	SIOP ESI
DoD Directive 5200.28	SOP
dominance	SPECAT
DSSCS	special access programs
Encrypted For Transmission Only (EFTO)	Standard Operating Procedures
For Official Use Only (FOUO)	TOP SECRET
format line	trusted
GENSER	UNCLASSIFIED
hierarchical	Unclassified EFTO
JANAP 128	window

GENSER MESSAGE SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)

Table 1 - Multi-Level Secure (MLS) Matrix

Trusted Operating System Levels	Classification	Categories					
			0	1	2	3	4
		/ (Default)	GENSER	GENSER_SIOPI_ESI	GENSER_SPECAT	GENSER_NATO	DSSCS
255	MAX						
70	TOP_SECRET	non-message data	message data	message data	message data	message data	
60	SECRET	non-message data	message data		message data	message data	
50	CONFIDENTIAL	non-message data	message data		message data	message data	
40	RESTRICTED					message data	
30	UNCLASSIFIED	non-message data	message data			message data	

1. Table 1 above identifies the *security levels* that a General Service (GENSER) message processing system will handle. Security level refers to two elements in combination, the *classification and the category*, and is held by the trusted operating system in the *sensitivity label*.

-- *Classifications* are hierarchical in nature and correlate to personnel security clearances, e.g. Top Secret.

-- *Categories* are non-hierarchical and are derived from special access programs that impose additional control requirements. For example, access to Top Secret SIOP-ESI requires a Top Secret clearance plus formal authorization for SIOP-ESI access.

-- *Sensitivity labels* are based on the classification and categories defined. This label is an additional piece of information that is attached to every object which the operating system controls.

The column "Trusted Operating System Levels" in the matrix shows hierarchical sensitivity levels within the trusted operating system. Notice that security levels are defined against operating system levels such that there is flexibility for the addition of security levels should that be necessary in the future. For example, "255" is the highest level in the trusted operating system, but the highest classification level is mapped to "70".

**GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)**

Table 1 shows the classifications, categories, and logical data combinations defined for a B1 multi-level secure operating system. In the matrix "message data" indicates the classification and category combinations that apply to JANAP 128 messages. Those that apply to other information (e.g., reference tables) and used during message processing/handling are indicated by "non-message data". The default category ("./") would be applied to this information since the information is not derived from a message. Blank cells in the matrix are illogical combinations that would never be used for data storage or user access, i.e. UNCLASSIFIED SIOP-ESI is not a legal message classification.

2. It is important to note that the hierarchical relationship of classifications enforces dominance of a higher level over a lower level, i.e. SECRET dominates CONFIDENTIAL, RESTRICTED, and UNCLASSIFIED. System users with access of SECRET also have rights to CONFIDENTIAL, RESTRICTED, and UNCLASSIFIED data within the categories included in the users' access. The default category is always available for the levels to which a user has access and so is not specifically included as part of the access or log-in. The non-default categories are exclusive and not hierarchical. That is, system access to GENSER_SIOP_ESI does not provide access to GENSER-NATO data. Several examples are shown in Table 2.

Table 2 - Data Access Allowed

Trusted Operating System Access	Message Data Access Allowed	Non-Message Data Access Allowed
SECRET//	none	SECRET// CONFIDENTIAL// UNCLASSIFIED//
SECRET /GENSER/	SECRET /GENSER/ CONFIDENTIAL /GENSER/ UNCLASSIFIED /GENSER/	SECRET// CONFIDENTIAL// UNCLASSIFIED//
SECRET /GENSER, GENSER_NATO/	SECRET /GENSER, GENSER_NATO/ CONFIDENTIAL /GENSER, GENSER_NATO/ RESTRICTED /GENSER_NATO/ UNCLASSIFIED /GENSER, GENSER_NATO/	SECRET// CONFIDENTIAL// none UNCLASSIFIED//

3. In a trusted windows environment using a Compartmented Mode Workstation (CMW) product, a window will reflect the security level at which the user logged in. The user will be able to display messages of security levels at or below the log-in security level. When a message is brought into a trusted window, the sensitivity label of the message itself *will not* be displayed. The security level of the process that initiated the trusted window will continue to

**GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)**

be shown as the window label. This should not be interpreted as a problem since the sensitivity label of the window will match or dominate the security level of the message.

As a matter of operational policy, users will have varying levels of clearances. A number of users, commensurate with site operational requirements, will be granted special access authorizations. The security level defined for each user will identify the classification level and also the categories authorized. Standard Operating Procedures (SOP) must be established for the translation of personnel security clearances to the trusted operating system access.

For example, the following clearances and accesses would translate to the trusted operating system access (log-on) as shown in Table 3.

Table 3 - Clearance/Access Translation

Security Clearance	Special Access Authorizations	Trusted Operating System Access
Top Secret	none	TOP_SECRET// (Personnel with TS clearance and no special access authorizations who require access to only non-message data.)
Top Secret	none	TOP_SECRET /GENSER/ (Personnel with TS clearance and no special access authorizations who require access to message data and non-message data.)
Top Secret	SIOP-ESI, SPECAT, NATO	TOP_SECRET /GENSER, GENSER_SIOP_ESI, GENSER_SPECAT, GENSER_NATO/ (Personnel with TS clearance and SIOP-ESI, SPECAT, and NATO special access authorizations who require access to message data and non-message data.)
Top Secret	NATO	TOP_SECRET /GENSER, GENSER_NATO/ (Personnel with TS clearance and NATO special access authorization who require access to message data and non-message data.)
Top Secret	NATO	TOP_SECRET /GENSER_NATO/ (Personnel with TS clearance and NATO special access authorization who require access to NATO message data, but not US message data, and to non-message data.)

Users assigned to correct messages may have Top Secret clearances and accesses of SIOP ESI, SPECAT, and NATO. Working in a secure area where Top Secret data could be openly displayed, users may sign on as:

TOP_SECRET /GENSER, GENSER_SIOP_ESI, GENSER_SPECAT, GENSER_NATO/

GENSER MESSAGE SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)

creating a window on the computer screen in which any message within the system could be edited. Alternatively, the same personnel working in a general service area not cleared for Top Secret materials and in which they would want to limit displays to Secret data and below, may sign on as:

SECRET /GENSER, GENSER_NATO/

creating a window on the computer screen in which only messages of Secret classification or lower, with or without NATO designation, could be accessed. That is, within this window neither a Top Secret message nor a Secret SPECAT message could be displayed. So, it is the trusted operating system access level assigned to personnel that determines the highest level at which they may log in to the system and it is the log-in that determines the levels at which they can access data during a particular session.

It would also be possible for users to create the two windows described above on the same computer screen. The trusted operating system would prevent copying of data from the higher level screen to the lower level screen - enforcing the read down/write up rule of trusted processing.

4. Application software will provide the processing necessary to determine JANAP 128 message classification. This includes validation of security information in Format Lines 2, 4, and 12 of the message and cross-check of Format Lines 2, 4, and 12. Additionally, special handling caveats in Format Line 4 (AAAAA or BBBBB) and Format Line 12 (SPECAT SIOP-ESI or SPECAT) are identified and validated if present. A trusted process, a distinct segment of software certified to raise and lower system privileges, will evaluate aggregate data and assign a sensitivity label that reflects the message security level.

5. In AUTODIN, *Unclassified EFTO* (Encrypted for Transmission Only) messages are handled through the use of the classification designator "E". These messages, although Unclassified, are considered sensitive and therefore they are transmitted over encrypted circuits only. Messages containing the classification designator "E", just as those containing the designator "U", will be assigned the UNCLASSIFIED/GENSER/ sensitivity label by the multi-level secure operating system. This is not considered a problem if all circuits connecting directly or indirectly to the message processing system are encrypted beyond the communications center.

6. The security level scheme depicted in the matrix of Table 1 is designed to accommodate potential interfaces with non-DoD/non-US multi-level secure messaging systems. The General Service (GENSER) category is defined to provide a mechanism for segregating NATO message traffic from other traffic (US and Allied).

**GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)**

7. The classification Restricted, used in AUTODIN and indicated by the classification designator "R", is not authorized for US originators. It may be used by an Allied or a NATO originator.

The handling of NATO Restricted and Restricted (indicates Allied originator) is different. NATO Restricted is handled as Unclassified For Official Use Only (FOUO). Restricted, from an Allied nation, is handled as Confidential.

In the processing system a message with the "R" classification designator will be assigned a sensitivity label as shown below:

Table 4 - Restricted Translation

Format Line 2	Format Line 12	Trusted Operating System Sensitivity Label
R	RESTRICTED	CONFIDENTIAL /GENSER/
R	ALLIED RESTRICTED	CONFIDENTIAL /GENSER/
R	NATO RESTRICTED	RESTRICTED /GENSER_NATO/

8. The *Defense Special Security Communications System "DSSCS" category* will not be used in a message processing system handling only AUTODIN GENSER information. The category is provided to accommodate potential interfaces to DSSCS systems or allow employment of the security level scheme in a DSSCS system.

9. Banners on a printed message will be provided by the trusted operating system. These banners will reflect the process sensitivity label identifying the classification and category, in that order. It is important to note that it is the process label and not the data label that will be printed. An application printer service, a trusted process, will evaluate all printing requests initiated from within the application and set the operating system print request to the same level as the data to be printed, ensuring consistent banners. Standard Operating Procedures (SOP) must be established for printing data directly from the operating system.

It should be recognized that DoD Directive 5200.28, Security Requirements for Automated Information Systems (AISs), enclosure 3, paragraph A.5. prescribes the following: "Automated markings on output must not be relied on to be accurate, unless the security features and assurances of the AIS meet the requirements for a minimum security class B1 as specified in DoD 5200.28-STD [the Orange Book]." This MLS scheme is designed for a B1, NSA-certified, operating system.

**GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)**

With the MLS schema presented, current classification markings would translate to sensitivity labels as follows:

Table 5 - Banner Translation

JANAP 128 Message Classification	Trusted Operating System Banner
COSMIC TOP SECRET	TOP_SECRET /GENSER_NATO/
SECRET	SECRET /GENSER/

**GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)**

A hardcopy of a message would have the following pages:

```
TOP_SECRET /GENSER_NATO/
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

#####   ####   ####   #####
#   #   #   #   #   #   #   #
#   #   #   #   #   #   #   #
#####   #   #   #   #   #
#   #   #   #   #   #   #
#   #   #####   #####   #

Request id: User-23      Printer: Printer-10

Wed Mar 1 09:10:12 1995

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

The sensitivity label of the user is:
TOP_SECRET /GENSER_NATO/
Unless manually reviewed and downgraded
The system has labeled this data:

TOP_SECRET /GENSER_NATO/

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
TOP_SECRET /GENSER_NATO/
```

Figure 1 - Banner Page of Printout

GENSER MESSAGE
SECURITY LEVELS (CLASSIFICATIONS & CATEGORIES)

```
TOP_SECRET /GENSER_NATO/  
  
data data data data data data  
data data data data data data  
data data data data data data  
data data data data data data  
data data data data data data  
data data data data data data  
data data data data data data  
  
TOP_SECRET /GENSER_NATO/
```

Figure 2 - Data Page of Printout

```
TOP_SECRET /GENSER_NATO/  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
  
The sensitivity label of the user is:  
TOP_SECRET /GENSER_NATO/  
Unless manually reviewed and downgraded  
The system has labeled this data:  
  
TOP_SECRET /GENSER_NATO/  
  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
TOP_SECRET /GENSER_NATO/
```

Figure 3 - Final Page of Printout

GENSER Message Multi-Level Secure (MLS) Classifications and Categories

Presented by
NCTAMS LANT
Mary Lou Hoffert
804-445-1808

NCTAMS LANT

CONCEPTS

- Multi-Level Secure (MLS)
- Sensitivity Labels
- Trusted Processes and Processing

ISSUES

- Applying MLS to
GENSER Message Processing
- Impact of Automated Processing
- US vs NATO Classifications

Table 1 Multi-Level Secure (MLS) Matrix

Trusted Operating System Levels	Classification	Categories					
		/ (Default)	0	1	2	3	4
		GENSER	GENSER_STOP_ESI	GENSER_SPECAT	GENSER_NATO	DSSCS	
255	MAX						
70	TOP SECRET	non-message data	message data	message data	message data	message data	
60	SECRET	non-message data	message data		message data	message data	
50	CONFIDENTIAL	non-message data	message data		message data	message data	
40	RESTRICTED					message data	
30	UNCLASSIFIED	non-message data	message data			message data	

SENSITIVITY LABEL

SECRET/GENSER,GENSER_NATO/

UNCLASSIFIED//

TOP_SECRET/GENSER_SIOPI_ESI/

ACCESS

- Data Access
- Clearance/Access Translation
- User/Process Access

Table 2 - Data Access Allowed

Trusted Operating System Access	Message Data Access Allowed	Non-Message Data Access Allowed
SECRET//	none	SECRET// CONFIDENTIAL// UNCLASSIFIED//
SECRET/GENSER/	SECRET/GENSER/ CONFIDENTIAL/GENSER/ UNCLASSIFIED/GENSER/	SECRET// CONFIDENTIAL// UNCLASSIFIED//
SECRET/GENSER,GENSER_NATO/	SECRET/GENSER,GENSER_NATO/ CONFIDENTIAL/GENSER,GENSER_NATO/ RESTRICTED/GENSER_NATO/ UNCLASSIFIED/GENSER,GENSER_NATO/	SECRET// CONFIDENTIAL// none UNCLASSIFIED//

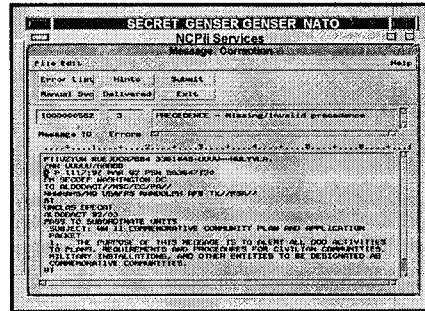
Table 3 - Clearance/Access Translation

Security Clearance	Special Access Authorizations	Trusted Operating System Access
Top Secret	none	TOP_SECRET// (Personnel with TS clearance and no special access authorizations who require access to only non-message data.)
Top Secret	none	TOP_SECRET/GENSER/ (Personnel with TS clearance and no special access authorizations who require access to message data and non-message data.)
Top Secret	SIOPI_ESI,SPFCAT,NATO	TOP_SECRET/GENSER,GENSER_SIOPI_ESI,GENSER_SPECAT,GENSER_NATO/ (Personnel with TS clearance and SIOPI_ESI,SPFCAT, and NATO special access authorizations who require access to message data and non-message data.)
Top Secret	NATO	TOP_SECRET/GENSER,GENSER_NATO/ (Personnel with TS clearance and NATO special access authorization who require access to message data and non-message data.)
Top Secret	NATO	TOP_SECRET/GENSER_NATO/ (Personnel with TS clearance and NATO special access authorization who require access to NATO message data, but not US message data, and to non-message data.)

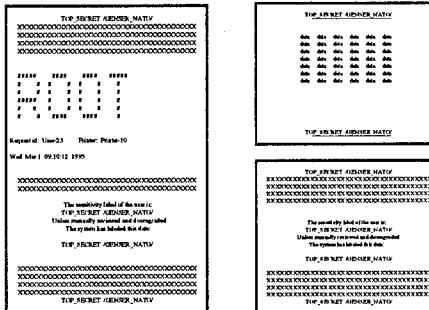
Table 4 - Restricted Translation

Format Line 2	Format Line 12	Trusted Operating System Sensitivity Label
R	RESTRICTED	CONFIDENTIAL/GENSER/
R	ALLIED RESTRICTED	CONFIDENTIAL/GENSER/
R	NATO RESTRICTED	RESTRICTED/GENSER_NATO/

Compartmented Mode Workstation - Screen Display



Printed Banners



CONCLUSION

- Leading Edge of Technology
- Accommodate Automated Processing
- Standard Nomenclature Benefits

A STANDARD AUDIT TRAIL FORMAT

Matt Bishop

Department of Computer Science
University of California at Davis
Davis, CA 95616-8562

Introduction

The central role of audit trails, or (more properly) logs, in security monitoring needs little description, for it is too well known for any to doubt it. Auditing, or the analysis of logs, is a central part of security not only in computer system security but also in analyzing financial and other non-technical systems. As part of this process, it is often necessary to reconcile logs from different sources.

Consider for example intrusion detection over a network. In this scenario, an intrusion detection system (IDS) monitors several hosts on a network, and from their logs it determines which actions are attempts to violate security (misuse detection) or which actions are not expected (anomaly detection). As some attacks involve the exploitation of concurrent commands, the log records may involve more than one user, process, and system. Further, should the system security officer decide to trace the connection back through other systems, he must be able to correlate the logs of the many different heterogeneous systems through whom the attacker may have come.

All this speaks of many needs, such as synchronization of time among hosts, a method for correlation of host-specific information, and a standard logging format. Such a format has several benefits. First, it makes analysis of the logs by a central engine simpler, because that engine need not know the types of systems generating the logs. Secondly, it enables logs generated for very different purposes to be reconciled. Suppose a credit card transaction is made over the Internet. The financial transaction will be logged at the (electronic) bank, and the connection (and presumably information about the transaction) at the purchaser's system. Should the purchaser claim fraud (e.g., he denies the transaction), the investigators would need to reconcile the system log with that of the financial institution to verify the legitimacy of the transaction. Third, it allows interoperability of audit systems on a very large scale, much the way a standard byte ordering allows interoperation of networked systems.

A standard log format robust enough to meet the needs of heterogeneity, transportability across various network protocols, and flexibility sufficient to meet a variety of needs in very different environments must satisfy two basic properties: extensibility and portability. Accepting existing log formats as standard violates one or more of these goals. For example, each of [4][5][7][8] are specific to a particular type of operating system, although the format described in [8] is meant to be general enough for third-party vendors to use. The format in [9] is specifically designed for the detection of misuse or intrusion in UNIX systems [6] and not for other situations such as financial transaction processing. Finally, the proposed POSIX standard [10] does not define a log format, but an application programming interface for accessing the log files a system produces. As the problem posed here includes moving the log files across networks and among heterogeneous platforms, use of such an interface in this context is inappropriate.

Extensibility implies that neither the names nor the number of the fields of the log record are

fixed. As the use of logs increases, investigators will become more sophisticated and demand additional information from the systems. Thus if the type of information that can be placed in the log record is limited to those quantities defined by the designers of the system, adding new fields requires a revision of the definition of an audit record as well as all ancillary software. Further, as designers become more sophisticated in what their systems will log, they will define new fields to aid in tracking specific security problems. All this speaks to allowing user-definable fields as well as common, predefined fields.

Portability implies that the log can be processed on any system. Thus, issues of byte ordering, character representation, and floating-point format must be either avoided or standardized. As log records may be sent over electronic mail, the format should be portable enough to pass through the SMTP protocol. This suggests that the best representation would involve printable ASCII characters only; note that canonicalizing the standard format to this requirement eliminates issues of byte ordering and floating-point representation, because numbers would be represented as ASCII strings, and the standard system conversion functions would translate these into numbers when required. Finally, given this approach, the record cannot be of fixed length, because different machines will have different precisions, and mandating that the ASCII representation of numbers be of a fixed length would potentially cause a loss of precision in some cases.

The next section presents our proposed format. In section 3, we show how and where the translation should be done, and in section 4 we demonstrate how log records from several disparate systems would be put into this format. Section 5 concludes with some observations and suggestions for future work.

Proposed Standard

We select as our goal the definition of a standard log record format. We explicitly do not attempt to standardize the events or fields (also called attributes) that are to be recorded; as argued in [3], that is more properly a function of policy and not of information interchange. Users of this format will have to use common field names when interoperating, and these common names could form the basis for another standard.

A log record consists of several fields all of which refer to the same event. We separate fields with a *field separator*, which by default will be '#'. (To include the separator in a field, repeat it; thus, "##" stands for a single '#' character.) Each field consists of an attribute, which is represented by a string of 1 or more characters not including '#' or '=', and a value, which consists of a string of characters; the two are separated by an '='. So, for example, the fields of a log record for a UNIX command may look like

```
#time=234627364#log=mab#role=root#UID=384#file=/bin#su#devno=3#inode=2343#
```

For the reasons stated above, log records cannot be of fixed length; they therefore require a start and a stop symbol. These symbols are pseudo-fields containing the characters "S" and "E"; note that these are not legal fields as they have no '=' in them. For simplicity, the special field "#N#" represents the juxtaposition "#E#S#". Thus, the above log record would be

```
#S#time=234627364#log=mab#role=root#UID=384#file=/bin#su#devno=3#inode=2343#E#
```

The SMTP protocol is quite restrictive; it requires that all characters be printable ASCII, and no line be more than 80 characters long. Hence, characters may, and nonprinting characters must, be represented by their value expressed in hexadecimal and surrounded by the *nonprinting delimiter* '\'. For example, if the value of attribute "controlchar" is "ESC-[H", where ESC is the escape character, the field would be

Figure 1. Summary of standard log format.

#S#	start log record	#Fc#	change field separator to <i>c</i>
#E#	end log record	#Cc#	change nonprinting delimiter to <i>c</i>
#N#	next log record (same as #E#S#)	#I#	ignore next field
#	default field separator	\	default nonprinting delimiter

\hex value represents the character with ASCII value *hex value*
attribute=value set the value of *attribute* to *value*

```
#controlchar=\1b\H#
```

This means that a ‘\’ character must be escaped, so the sequence “\\” represents a single ‘\’. Further, a mechanism for including newlines in the middle of a log record will allow the record to be broken into lines of less than 80 characters; for this purpose, we define the pseudo-field “#I#” as marking the next field to be ignored. (Incidentally, this also allows comments to be interpolated.) To expand on our log record above:

```
#S#login_id=bishop#role=root#UID=384#file=/bin/su#devno=3#inode=2343#I#
#return=1#errorcode=26#host=toad\79\#E#
```

As one last feature, we note that the field separator and the nonprinting delimiter may occur often in the value of fields on some systems. Hence, we provide a way to change both. The distinguished symbol “#F%#” changes the field separator to ‘%’ and the symbol “#C\$#” changes the nonprinting delimiter to ‘\$’. Note that any character may be used, not just ‘%’ and ‘\$’. Also note these are illegal fields as there is no ‘=’ in them. For example,

```
#S#F%#C$#login_id=bishop%role=root%UID=384%file=c:\bin\load%I%
%return=1%errorcode=26%host=toad$79$%E%
```

Note that these symbols are not considered part of the log record in which they occur; rather, the chosen field separator and nonprinting delimiter characters remain in effect until changed. Figure 1 summarizes these character sequences.

Note that we do *not* specify any particular attributes as standard. This is to allow the designers of audit tools to name fields as they wish; so long as they are consistent across platforms being audited, the precise names of the attributes do not matter. However, many systems log the same categories of information (such as user name, command, date, and process number). Section 4 describes several such attributes, and names and representations are suggested.

Note also that this format eliminates the problem of the undefined value. In a system in which some attributes are required, the log must be able to specify that the value for the attribute cannot be determined. Here, one of two approaches may be taken. First, define a distinguished value to represent the undefined value; this is the approach other log formats use. Second, simply omit the attribute from the record. If it is not present, then it is clearly not defined. This approach eliminates the need for a distinguished value to mean undefined.

Use of the Format

Because each system uses its own internal representation of log files, and its own auditing tools are crafted to use that format, it is not necessary that the log records be put into the standard format. The need for a standard format arises when tools recognizing only that format are used, or when the logs generated by that system must be combined with logs from other, different types of,

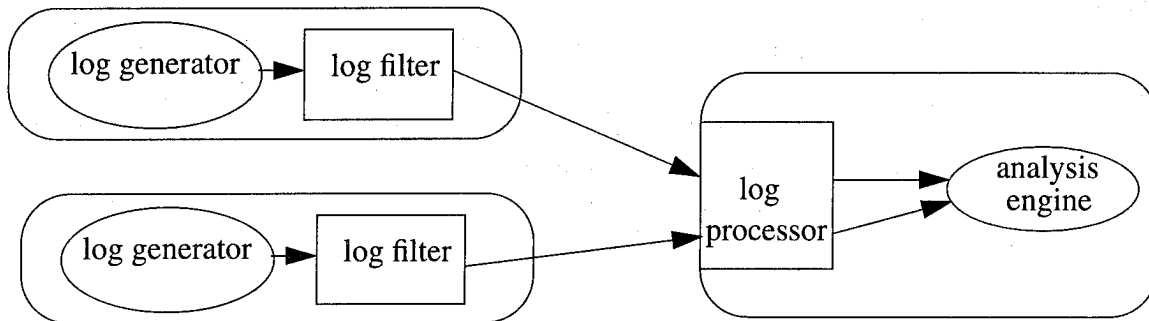


Figure 2. Architecture using log filters to generate the standard format. The native system logs (generated by the log generators) are translated into the standard format by the log filters and then sent to the log processor on the analysis host. That program changes the standard format into the internal representation used by the analysis engine.

systems.

Hence the recommended architecture for generating this log format is to build a filter tool that will take as input the raw log records as produced by the system, and will generate as output the standard log format. With this approach, at the analysis engine one need write all log input programs to use only the standardized format. Figure 2 summarizes this approach.

We note that one could use the POSIX standard interfaces to define the manner in which the filter should access log records. In this case, the API would be the same for all POSIX-compliant machines and the processing of the information would vary. We note however that the POSIX interface suffers from some limits, specifically a failure to include some relevant information such as session identification mechanisms, and that few vendors provide POSIX-compliant interfaces.

This approach avoids the need to modify the kernel locally if new information becomes available. For example, suppose initially the log only records the user time of a process, and a later revision adds system time spent executing on behalf of the process to the log. The filter will now need to be changed to add this information into the standard format log, but the operating system need not be modified (beyond the upgrade).

As an aside, we note that the filter may reside on the system being monitored (in which case the records will be sent in standard format) or on the analysis engine (in which case the logs will be sent in native format). The former seems preferable because it not only distributes the computation load but also handles network dependencies such as network byte order.

The next section presents several issues about representation of values to demonstrate complexities that arise in using this format. We do so by examining log records for several systems.

A Comparison of The Standard Log Format with Other Formats

In this section, we describe several log record formats, and show how they can be mapped into the standard audit format.

Basic Security Module

The Basic Security Module (BSM) [4] is an enhancement to SunOS system security. Each log record is made up of a sequence of tokens and, like the standard format, the record size is not fixed; there is a begin and an end token. Each record refers to an auditable event, which may be a “kernel event” such as a system call or an “application event” such as a failure to authenticate suc-

cessfully to the login program.

BSM defines a token to be a token identification field followed by a series of information fields. These tokens all relate to user identity (process, which includes real, effective, and original UID and effective group ID as well as process ID; group list), file system information (pathname and attributes), IPC usage (IPC token, IPC attributes), networking (IP port number, IP address), and process and system call information (return value, arguments) as well as more general information (text, data, opaque). By using this information, actions on the system can be traced.

The BSM logs use the same free-format idea as the standard log format; the only differences are that the BSM information is stored in binary format when appropriate (for example, if numbers are involved) and the start and end tokens contain the length of the record. The standard log format does not do this to allow the records to be generated on the fly, so that the entire record need not be constructed in memory and then output. This means that scanning the standard log format may involve some overhead, but the overhead is most likely negligible and is offset by the elimination of the need to process ASCII strings into numbers.

An example BSM log record might look like this (when formatted using *praudit*):

```
header,35,AUE_EXIT,Wed Sep 18 11:35:28 1991, + 570000 msec,  
process,bishop,root,root,daemon,1234,  
return,Error 0,5  
trailer,35
```

Put into the standard log format, this looks like:

```
#S#event=AUE_EXIT#date=09181991@113528#usedtime=570000#logid=bishop#I#  
#ruid=root#euid=root#egid=daemon#procid=1234#errno=0#retval=5#E#
```

Note that the same information is present, but the attributes are named rather than defined by location in the log record. This is necessary as different systems and different policies will require different information to be stored, leading to much confusion if the fields are not identifiable by attribute name rather than position. Basically, one cannot predict all attributes that will need to be logged; hence, one cannot rely on position.

SunOS MLS Logs

SunOS MLS, the multilevel secure version of SunOS, produces logs very similar to those of the BSM [8]. Log records are not fixed length, but there is no trailer token; the header token includes a length, type, and time field. Associated with each event is a header token, a subject token (giving the login, real, and effective UID and real GID of the process and the associated user), return value information, labelling information (if the system uses labels), and other ancillary information identical to that of the BSM. The average size of a log record is between 120 and 180 bytes; compression reduces this appreciably (by roughly a factor of 4 to 8, depending on the record's contents).

A simplified example of a SunOS MLS log record is given in [8]:

```
header,120,AUE_UNLINK,Wed Sep 18 11:35:28 1991, + 570000 msec,  
process,bishop,root,root,daemon,1234,  
label,confidential,nuclear,crypto  
pathname,/,/usr/holly,..matt/tmp/junkfile  
return,Error 0,5  
trailer,120
```

Put into the standard log format, this looks like:

```
#S#event=AUE_UNLINK#date=09181991@113528#usedtime=570000#I#
```

```
#logid=bishop#ruid=root#euid=root#rgid=daemon#procid=1234#I#
#secllevel=confidential#class=nuclear#class=crypto#I#
#rootdir=#cwd=/usr/holly#pathname=../matt/tmp/junkfile#I#
#errno=0#retval=5#E#
```

Again, note the standard log format simply presents the information in another way. Also note that if the attribute names are too long, one could define very short ones.

The basic differences between this format and the standard log format are twofold. First, SunOS MLS log records include data in integer format; second, the types of information that can be placed in those records is constrained and not easy to change. For example, if the same format were used on a financial system, the format would need to be changed to include information about the transaction itself. However, this format is quite good for its intended purpose (which is to provide information for system security auditing).

VAX VMM Security Kernel

The VAX VMM security kernel is a virtual machine monitor which has extensive auditing abilities designed to meet the requirements of the A1 class of the Orange Book [7]. All logging is done by the Audit Trail layer and each record contains an event identifier, the event status (result of the event), auxiliary data (such as the name, type, and class of the object involved in the event, and other event-specific information), the name of the caller (who caused the event), the date and time of the event, the caller's type, access class, user's name, rights, and privileges. While some events can be excluded from the log, the higher layers have the power to override exclusion (for example, if a login fails, the event will be logged). Unfortunately, the paper gives no examples, but the attributes here can clearly be captured by the standard log format.

Again, this format has some drawbacks as a standard log format: the attributes are fixed, and the data in the logs is binary, so numbers (for example) are stored in a machine-dependent manner. To be fair, it was intended only for use in the VAX security kernel, and for that purpose appears to be quite good.

svr4++ UNIX Log File Format

This log format [9] is an ASCII format based on the logging format used in OSF/1. The attributes entered in a log record are time, event type, process identifier, result, user and group information, session identifier, labelling information for the process, information about the object (name, type, security label, device and inode information) and miscellaneous data. Each log record is a single line with comma-separated fields, and undefined fields (such as the security label field when the process does not have a security label) are set to '?'.

This style of record approaches portability. It is in ASCII, which solves the problem of binary data management. However, the fields it uses are tied directly to the nature of the policy which suggested the creation of the log: misuse or anomaly detection. No extensibility is provided for (the miscellaneous fields are labelled as being dependent on the operating system and the event).

Here is an example audit record in this format (it is spread over two lines for clarity):

```
16:36:01:28:09:92,6,P16195,s(0),1021:1021:1021,10,S?,?,
(/home/snapp/creat.foo:f:"0644,1024,10":17080:66:184:411265:1818)
```

The equivalent record in the standard log format is:

```
#S#event=6#date=09281992@163601#logid=1021#ruid=1021#euid=1021#rgid=10#I#
#procid=16195#objname=/home/snapp/creat.foo#objtype=file#objmode=0644#I#
#objuid=1024#objgid=10#objdevid=17080#objdmaj=66#objdmin=184#I#
```

#objino=411265#objfsid=1818#E#

A few remarks are in order. First, had multiple objects been present, the attributes could be numbered obj1..., obj2... and so forth to distinguish the object to which the fields referred to. Secondly, this log record assumes that the audit engine knows the internal representation of users (for example, that user id 1021 refers to John Smith). Third, the label field and session id field are omitted as the values in the svr4++ log record fields show the system did not provide those. This makes the log more readable.

A Log for an Embedded Avionics System

The study of log records for an avionics system [5] may seem far from the point of this paper, but as we claim the format is general enough for all purposes, this serves as one way to test our claim. The log records subject identifier, action performed, 2 security-relevant parameters, object identifier, the initial and resulting value of the object and the status of the operation, and then information about resource usage, a time stamp, and the severity of the event and the status of the logging. Again, the paper gives no examples, but clearly the standard format provides enough flexibility to allow the records to be standardized.

RACF

RACF [1] is a security enhancement package for the IBM MVS operating system and VM environment. It logs failed access attempts and the use of privileges to change security levels, and can be set to log any RACF command, changes to the RACF database, attempts to access resources guarded by RACF, and any access by privileged groups or users. The logged information includes userid, name, owner of the resource, when the resource was created, and so forth.

RACF generates reports using four commands. LISTUSER lists information about RACF users:

```
USER=EW125004  NAME=S.J.TURNER  OWNER=SECADM  CREATED=88.004
  DEFAULT-GROUP=HUMRES  PASSDATE=88.004  PASS-INTERVAL=30
  ATTRIBUTES=ADSP
  REVOKE DATE=NONE  RESUME-DATE=NONE
  LAST-ACCESS=88.020/14:15:10
  CLASS AUTHORIZATIONS=NONE
  NO-INSTALLATION-DATA
  NO-MODEL-NAME
  LOGON ALLOWED      (DAYS)  (TIME)
-----
  ANYDAY              ANYTIME
  GROUP=HUMRES  AUTH=JOIN  CONNECT-OWNER=SECADM  CONNECT-DATE=88.004
  CONNECTS=    15  UACC=READ  LAST-CONNECT=88.018/16:45:06
  CONNECT ATTRIBUTES=NONE
  REVOKE DATE=NONE  RESUME DATE=NONE
  GROUP=PERSNL AUTH=JOIN  CONNECT-OWNER=SECADM  CONNECT-DATE:88.004
  CONNECTS=     25  UACC=READ  LAST-CONNECT=88.020/14:15:10
  CONNECT ATTRIBUTES=NONE
  REVOKE DATE=NONE  RESUME DATE=NONE
  SECURITY-LEVEL=NONE SPECIFIED
  CATEGORY AUTHORIZATION
  NONE SPECIFIED
```

A standard log format representation of this might be:

```
#S#user=EW125004#name=S.J.TURNER#owner=SECADM#created=01041988#I#
```

```
#defgroups=HUMRES#passdate=01041988#passinterval=30#attributes=ADSP#I#
#lastaccess=01201988@141510#logonok=anyday,anytime#group1=HUMRES#I#
#group1auth=JOIN#group1connowner=SECADM##group1conndate=0104995#I#
#group1conncount=15#group1uacc=READ#group1lastconn=01181988@1641506#I#
#group2=PERSNL#group2auth=JOIN#group2connowner=SECADM#I#
#group2conndate=0104995##group2conncount=25#group2uacc=READ#I#
#group2lastconn=01201988@141510#E#
```

The other three log formats may be translated similarly. Note the difference in attribute names which reflects the difference in security policy and system implementation.

CA-UNICENTER

CA-UNICENTER is a UNIX-based product providing many security features of a mainframe. Its log messages cover logging in, logging out, and resource protection. Among the attributes recorded are event, login name, host name, terminal identifier, resource name, result, and access request. For example, the CA-UNICENTER record

```
CASF_E_465 Access violation by bishop to asset (Warn) /bin/su> from source console for access type write
```

would be

```
#S#event=CASF_E_465#loginid=bishop#mode=Warn#asset-name=/bin/su#I#
#termid=console#reqaccess=write#E#
```

in the standard log format. Similarly, the record

```
CASF_E_466 Logging access by bishop to asset /bin/su from source console for access type execute
```

would be translated to

```
#S#event=CASF_E_466#loginid=bishop#asset-name=/bin/su#termid=console#I#
#reqaccess=execute#E#
```

Summary

We have taken examples of log records from very different systems and shown how to put them into the standard log format. This demonstrates that the log format can handle a variety of systems and security policies, from intrusion detection to financial records.

We should note some commonalities between the attributes in the different examples. First, user ID may be represented either by name or number, but the analysis engine must be able to resolve either to a canonical name. The representation of date and time is as *mmddyyy@hhmmss* rather than as an internal number (such as the number of seconds since January 1, 1970) because different systems use different numbers, so this was chosen to make the records easier to understand. Of course, all systems must have synchronized clocks to make a comparison of times meaningful.

Example Attack Record

In this section we suggest specific fields for system security; that is, what fields in the standard log format would a security analyst trying to track an intruder find useful? A fully detailed analysis would be beyond the scope of this paper, but a simple one follows.

Intruders enter systems through a variety of mechanisms, most involving network connections or logins. (Note that an exception is piggybacking onto an active connection.) Hence, log fields indicating the origin and type of connections are appropriate, as is the time and privilege of the connection. For reference, each log entry should be numbered. For example:


```
#S#no=1231#date=09281992@163601#net=1#srv=smtpd#orig=123.45.67.89#port=25#E#  
#S#no=224#date=10101997@123456#tty=console#usr=mab#role=mab#grp=fac#tryno=1#E#
```

The first line is an example of an SMTP connection originating from IP address 123.45.67.89 and coming in over the first network (this is a multi-homed host), and the second a login by user "mab" from the terminal "console"; the login was successful on the first try, and the user was put into group "fac" and the role "mab".

Detection involves looking at commands executed; the axes here are for suspicious programs (such as a user executing a program called "guess_anyones_password"), and normal programs that deviate from their expected pattern of execution (such as a UNIX shell with 100 hours of CPU time; shells virtually never have that much CPU time). Fields relevant here would be program name, amount of execution time (system and user, as well as time of execution, termination, suspension, and resumption), and files accessed. Some sample log entries are:

```
#S#no=123#name=/bin/sh#date=10101997@123456#act=begin#usr=mab#grp=fac#I#  
#cwd=/u/mab#arg1=X#pid=9876#E#  
#S#no=124#name=sh#date=10101997@123457#pid=9876#act=open#mode=read#I#  
#usrtime=0.01#systemtime=0.01#file=/u/mab/X#res=1#I#  
#fdev=/dev/rrh0e#fino=123214#ftype=reg#fperm=0644#fuser=mab#I#  
#fgrp=fac#atime=10081997@102300#ctime=080396@153451#I#  
#mtime=10021997@023534#E#
```

In the first line, the program "/bin/sh" (process number 9876) has been started by user "mab", in group "fac", and was given the argument "X". The second entry shows that "/bin/sh", with 0.01 seconds of user and system time on it so far, has tried to open file "/u/mab/X" for reading and has succeeded. The file resides on device "/dev/rrh0e" with inode number 123214, has access permissions "0644" (owner read and write, group and other read), and is owned by user "mab" and group "fac". It was last accessed at 10:23:00 on 10/8/1997, last modified at 2:35:34 on 10/2/97, and created at 15:34:51 on 8/3/96.

```
#S#no=139#name=/bin/sh#date=10101997@125001#pid=9876#usrtime=21600#I#  
#systemtime=0.01#act=susp#usr=root#E#  
#S#no=160#name=/bin/sh#date=10101997@125223#pid=9876#usrtime=21600#I#  
#systemtime=0.01#act=term#usr=root#E#
```

These last two log entries show that process 9876, which is "/bin/sh", was suspended and subsequently terminated by user "root". At the time of suspension, it had 21600 seconds (6 hours) of user time and 0.01 seconds of system time.

This is a very simple example of what a system administrator would look for. Note that in this example the user time and system time are written out at each system call. Whether or not all these fields could be present depends on the system on which the logging is done; but there is no question their presence would indeed be useful.

Conclusion

This paper has presented a very flexible, portable, extensible standard log format. We have demonstrated its use by applying it to several different formats of log records.

The key issue is, of course, what to log. As shown in [3], what to log depends on both the implementation of system logging mechanisms and the needs of the security policy to be enforced. This paper speaks to neither point; nor does it claim to.

The architecture of a distributed auditing system is beyond the scope of this paper, but the essentials of one such system are described in [2]. That paper does not deal with reconciliation of

logs from heterogeneous systems, which is a very deep research question. This paper presents work that is a step in the direction of a solution by eliminating the need to have the reconciliator understand the vendors' log format. The next step is to investigate techniques to reconcile logs.

Acknowledgments: Thanks to Al Novissimo and Alan Paller of Computer Associates, Inc., for providing information about CA-UNICENTER, to John Gregg and David Day of the UC Davis Office of Internal Audit for useful discussions about non-computer oriented auditing involving the analysis of computer systems and for a description of RACF, and to Biswaroop Guha, Christopher Wee, James Hoagland, and Karl Levitt for useful discussions. This work was supported by an award from the Lawrence Livermore National Laboratory to the University of California at Davis, and was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48

Basic Security Monitor and SunOS are registered trademarks of Sun Microsystems, Inc. CA-UNICENTER is a registered trademark of Computer Associates, Inc. RACF is a registered trademark of IBM. VAX is a registered trademark of Digital Equipment Corporation.

References

- [1] *Audit, Control, and Security Issues in RACF Environments*, Technical Reference Series No. 37052, Ernst & Whinney; available from The EDP Auditors Foundation, Inc., Carol Stream, IL (1992).
- [2] D. Banning, G. Ellingwood, C. Franklin, C. Muckenhirn, and D. Price, "Auditing of Distributed Systems," *14th National Computer Security Conference Proceedings* pp. 59-68 (1991).
- [3] M. Bishop, "Goal-Oriented Auditing and Logging," *unpublished*.
- [4] *Installing, Administering, and Using the Basic Security Module*, Sun Microsystems, Inc., Mountain View, CA (April 1992).
- [5] K. N. Rao, "Security Audit for Embedded Avionics Systems," *Proceedings of the Fifth Annual Computer Security Applications Conference* pp. 78-84 (Dec. 1989).
- [6] D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," *Communications of the ACM* **17**(7) pp. 365-374 (1974).
- [7] K. F. Seiden and J. P. Melanson, "The Auditing Facility for a VMM Security Kernel," *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy* pp. 262-277 (1992).
- [8] W. Olin Sibert, "Auditing in a Distributed System: Secure SunOS Audit Trails," *11th National Computer Security Conference* pp. 81-91 (1988).
- [9] Stephen E. Smaha, *svr4++*, *A Common Audit Trail Interchange Format for Unix*, Haystack Laboratories, Inc., Austin, TX (Oct. 5, 1994).
- [10] *Standard for Information Technology Portable Operating System Interface (POSIX) Part I: System Application Program Interface (API)*, Report 1003.1e, (April 1994).

TCP/IP (Lack of) Security

Jesper M. Johansson

BMGT 727

Security of Information Systems

Dr. John Campbell

Abstract

This paper explores the security problems of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. The security problems of many of the most common features are explained, and examples are given in many cases. The paper also takes a look at the changes to the security aspects of IP which will change with the adoption of the revised version of this protocol: IPv6. This protocol has solved some of the security problems inherent in IPv4, but many problems, especially those that are inherent to other areas of the protocol, and those which rely on source address authentication, remain. The paper concludes by a short examination of what was perhaps the largest security breach in IP history, the Internet Worm. What the worm actually did and did not do will be covered, as well as how it operated. This is an important exercise, since it highlights some of the major security flaws in the protocol suite. It also highlights the dangers of allowing the users of the system, as opposed to the system manager, to dictate security policy.

Acknowledgment

I would like to thank my professors at the University of Maryland, and especially Drs. Campbell and Alavi, for allowing me the opportunity to explore many of the areas of Information Systems which I normally would never have delved into. When I started with the program I had no idea how wide and exciting the area is. You have made me realize the excitement in studying the leading edge of the field. It is due to your encouragement and advice that I have decided to make the exploration of the mysteries of Information Systems my life work. As I commence my doctorate studies at the University of Minnesota, I am forever grateful to you for your guidance and encouragement.

TCP/IP (Lack of) Security

The TCP/IP protocol suite is arguably the most commonly used protocol suite in the world today. It comes as a standard feature on virtually all UNIX^{®1} systems. It forms the base of the largest network in the world: The Internet. The Internet was developed using grants from the Department of Defense's (DoD) Advanced Projects Research Agency (ARPA). The Internet, and the TCP/IP protocol suite it is built on, were not designed to provide security features. (They also were not designed to handle the number of hosts presently on them either, but more about that later). Rather, they were designed to facilitate the dissemination of information. Therefore, most security features were added as an afterthought, in higher level protocols.

This paper discusses the security aspects of TCP/IP. First, a relatively brief overview of the Internet Protocol (IP) will be given. This discussion will also cover the layering of protocols, to create the protocol stack. Next, the "auxiliary" protocols (the higher layers) will be presented, along with the security features they provide, or, as in most cases, lack. We will then turn our attention to hope for the future: Internet Protocol; The Next Generation, more commonly known as IPng, or officially IPv6 (as opposed to IPv4, the current implementation). Lastly, we shall delve into a case study on the dangers of trust: The Internet Worm of November 1988.

The Internet Protocol

The Internet Protocol (IP) was first developed along with the ARPANet, and has been included in the release of most every UNIX implementation since Berkeley's version 4.2 (BSD4.2). The Internet protocol provides the equivalent of the Open Systems Interconnect (OSI) networking layer to the TCP/IP stack, with the exception that, in contrast to OSI, TCP/IP actually works. TCP/IP is both topology and data link independent. This is one of the strong points, since the protocol thus can be utilized on almost any network [KEEN94]. There are also numerous derivative protocols of IP, such as IPX (used in Novell Corp's LANs). IP, however, has significant shortcomings, other than the security ones, which we shall discuss soon. First, we will discuss some of the basics of IP though.

IP is a packet switched protocol. This means that it divides the message to be sent into packets, which then could take different routes to the destination. The destination, and the source, is identified through their IP-addresses. An IP address is a 32-bit number, where the first few bits define what size network the host is connected to. IP does not check whether a packet was received or not. If a packet is undeliverable, or does not reach its destination, it is simply discarded. Higher level protocols deal with what to do when packets disappear. An IP packet consist mainly of two parts: The header, and the datagram. The header specifies the source, the

¹Unix is a trademark of AT&T Bell Laboratories

destination, and certain other information, as will be described later. The datagram carries the actual message, in plaintext. There is no method for encryption currently available that works on the IP level.

IP Shortcomings

The most notable shortcoming of IP is the fact that its address space is fairly limited. An Internet address (see Table 1) is made up of a 32 bit number. That number is further subdivided into (1) high-order bits, specifying the class of network; (2) the network portion, specifying actual network within the class; and (3) the host portion, specifying the host within the network.

IP Address Formats					
Class	High-Order Bits	Network Portion	Host Portion	Number of Networks	Number of Hosts
A	0	7	24	128	16,777,214
B	10	14	16	16,384	65,534
C	110	21	8	2,097,157	254
D	1110	Multicast Group	Multicast Group	-	268,435,456
E	1111	(Experimental Use)	(Experimental Use)	-	-

Source: [CHES89]

The class A networks have long been exhausted. The Class B networks were already in 1990 estimated to be exhausted by March of 1994 [BRAD95]. The Internet thus faces significant growing pains. The theoretical limit on the number of hosts on the Internet would be just

under 4 billion. However, despite this limit, the routing tables necessary to implement a full scale network with all these hosts would be humunguous, thus significantly slowing performance. Therefore, an upgrade to the current IP protocol is being developed. This upgrade is discussed at more length in the IPv6 section of this paper. It is estimated that, under the current conditions, and rate of issuance, the address space will be exhausted between the years 2005 and 2011 [BRAD95]. However, this projection does not take into account any possible significant shifts in the rate of usage.

Another weakness, which is more directly related to this paper, is the lack of security features in IP. There are a number of optional fields in an IP header, two of which are the security label,

and the strict and loose source routing fields. There are no other security features inherent in the current version of IP.

The Security Label

The Security Label is most commonly used in military applications. The field allows a packet to be labeled with the sensitivity of information it contains [CHES94]. These labels follow the well-known "military model." However, most operating systems in use today make no use of these labels. The most common current use for them is to restrict routing. For example, a packet labeled Top Secret will not be transmitted over a router rated less than that, unless the packet is properly encrypted, using Top Secret-rated keys.

Source Routing

Source routing is an option in IP which specifies the routing path that a packet should take. This is not a security feature in IP. Rather it is a significant problem. Using this label, a person can specify that a packet should take a certain route. Since the destination machine must use the inverse of that route [BRADEN89], the attacker can impersonate any machine that the target trusts, thus gaining access to the packets, as they pass by. It is generally recommended that source routing is turned off, or that packets containing this option be rejected by the router. There are very few legitimate uses for source routing.

Higher Level Protocols

There are a number of auxiliary, or higher-level, protocols which attach to IP. By itself, IP will do nothing more than establish a connection to another computer. It is up to the higher level protocols to decide what to do with that connection. Many of these protocols suffer from the lack of basic security features in IP. Most of them also provide some holes all of their own. These protocols can do many different things, ranging from setting up and tearing down connections to other computers (TCP), to sending mail from one computer to another (SMTP).

Connection Layer

TCP

The most well known of the higher level protocols is the Transport Control Protocol (TCP). This protocol is so commonly mentioned in conjunction with IP, that most people probably think that they are one and the same. This is far from the truth. TCP is not required to do much of the work on the Internet. There are other protocols available to do some of the same things.

TCP is used to provide the user (whether a breathing human being or a process on a computer) with a reliable virtual circuit to use for the communication. (Interestingly, TCP adds a connection-oriented protocol to a packet switched network, a seeming paradox). Since IP is packet switched, there is no guarantee that packets are in the right order when they arrive. TCP takes care of this by ordering the packets according to their sequence numbers. Every byte sent

carries such a number. This sequence number is also used as an acknowledgment number, sent the other way, to signify the last successful packet sent. Every packet, except the first one sent, will contain such an acknowledgment number. The initial sequence number (at least in a Berkeley system) is incremented by a constant amount each second, and by half that amount, each time a connection is initiated [BELL89]. Robert T. Morris pointed out that it is entirely possible to predict that sequence number, thus fooling the attacked host into thinking that someone else is connecting to it [MORR85]. A normal TCP connection sequence is shown in Figure 1. The client would send a SYNchronize message to the server, including an initial sequence number (ISN). The server would respond with its own synchronization and an acknowledgment of the clients ISN. The client would then acknowledge the server's ISN and the data transfer could start. This is commonly known as the three-way handshake in TCP.

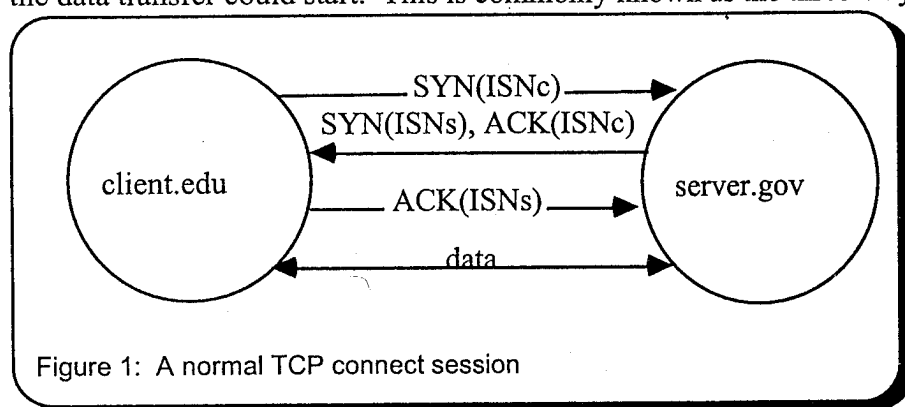


Figure 2 pictures an attack. First the attacker would have to select a host to use as a third party. This is the host to be blamed for the attack. Preferably, this host should be a host trusted by the victim. The attacker would flood that third party with connection requests. This would cause that host to be unable to respond to communication from the victim. The attacker would then initiate a TCP session to the victim, stating that s/he is in fact the trusted third party host. The victim, believing that the address sent as the source, is in fact the source, would send the usual acknowledgment to the third party. However, since the third party is busy, the packet would simply be thrown away. Since no errors are generated, the victim still does not know that it is under attack. The attacker would now, assuming that s/he can reasonably guess the ISN that the victim sent to the trusted host, acknowledge this ISN. This establishes the connection. The victim still thinks that the third party is in fact the one connected, and a normal session would proceed. The attacker could now access anything at the victim's site that the third party has access to. This type of attack, which is known as a sequence number attack, would essentially circumvent the only real authentication feature in TCP: Source address authentication. This authentication method is based on the fact that a host trusts other hosts based on the source address that they supply it with. Many commands rely on this authentication feature, most notably the Berkeley r-services (discussed later). This forms the basis of lesson one in internetworking: Trust no-one. The only method of authentication today is the source address

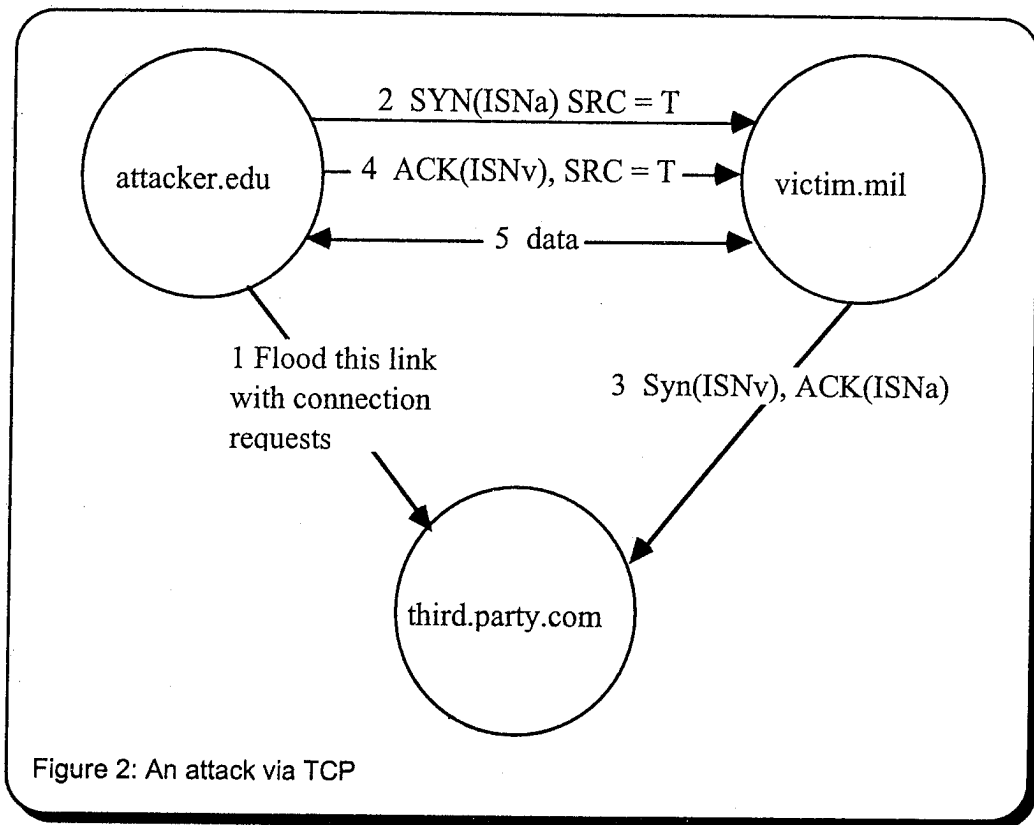


Figure 2: An attack via TCP

authentication. However, seeing how easy it is to spoof source addresses, one cannot be certain that someone connecting is who s/he says s/he is. This point is driven home further in the discussion of SMTP later. Note that, if the victim had blocked legitimate connections, such as via a firewall, this method of breaking in will not work. However, firewalls will be the subject of an entirely different paper. Another way to foil such an attack is to create more random (read “not guessable”) initial sequence numbers, like for example, random bits from RAM. However, the TCP specification does not provide for this. Rather, it provides for the fact that the number be changed at a rate of 250 hertz. Since the spoofer could probably predict the constant increment to the ISN, if s/he could only measure the turnaround time from an initial probe, all the guesswork is taken out of spoofing [BELL89].

UDP

The User Datagram Protocol (UDP) provides additional security problems. UDP is a transport protocol that extends the service of IP to applications. As such, it has no guarantees of service. When using UDP, there is no setup and tear-down of a circuit as with TCP. Rather, UDP simply transmits the packet, and hopes that the host is responding. This makes it very suitable for query/response applications, where there is no need to setup a virtual circuit. Since there is no handshake, there is no way to authenticate the sender of a UDP packet. Therefore, the application using UDP will have to provide some form of authentication service.

ICMP

The Internet Control Message Protocol (ICMP) is used to inform hosts of different, and ostensibly better, routes to other hosts. In addition to this, it also supports a program called ping which is used by system administrators to monitor systems. The problem with ICMP messages lie in that many older implementations do not use the service correctly [CHES94]. When a message is received that some host was unreachable for a specific connection, these older implementations will disable all connections to the other host. Thus, sending false ICMP messages could prove a very effective denial of service attack. Also, an ICMP message can be generated by a prospective attacker, informing the victim that the usual path to a specific host is down, and that a different path should be taken instead. Packets could then be sent via the attackers machine, where they can be conveniently read, altered, or plain lost. In order to safeguard against this, ICMP redirect messages should not be obeyed unless they come from a router that is directly attached to the victims network [BELL89].

The Daemons

Daemons are programs which provide services on a UNIX machine. An example of a daemon is fingerd which provides the finger service. This service is used to look up all kinds of useful information about a user. Services such as these are usually used by crackers to obtain information about users, that can later be used to determine login names and guess passwords. Many of the attacks on these services center on a few files, such as /etc/passwd, /etc/hosts.equiv, and \$HOME/.rhosts. These files, respectively, provide information on: All of the users authorized on a system, their authorization level, their encrypted password, etc; hosts trusted by the current system; and hosts trusted by individual users. We will now discuss some of these services in more detail.

SMTP

SMTP stands for Simple Mail Transfer Protocol. Virtually all system administrators, whether connected to the Internet or not, state that e-mail is the most sought-after service there is. The first problem when using SMTP and the Internet is not really a security problem: SMTP supports 7-bit ASCII (American Standard Code for Information Interchange) communications. This means that anyone using the Internet for communication in a language other than English will have a problem: SMTP does not recognize any characters with an ASCII number above 127. This precludes it from using characters used commonly in languages such as French, Spanish, German, and Swedish, not to mention the two-byte languages (Chinese, Japanese etc). However, there are much more serious problems with SMTP.

The most apparent problems in SMTP is that there is absolutely no way to be sure of who sent a message. As a matter of fact, the SMTP daemon does not even check that the domain or user name that the message is purported to originate from exists! (The author has been able to send messages to himself from both long since closed accounts, and fake domains). This bug (as it

almost has to be called) is also present in many third party add-ons to SMTP. For Example, Eudora, a very popular program used as a front end for a Post Office Protocol (POP) gateway on a Macintosh, does not even bother to log in to the POP server before sending messages. It is thus possible to send mail without even having an account, anywhere! There are many other programs with similar features, and the same lack of any attempt at authentication.

However, while people who are even vaguely familiar with the Internet should be aware that there is no guarantee that the message actually originates from the user it says, there are much more serious flaws in SMTP. One such flaw has been extensively reported (see SPAF89, EICH89, ROCH89 etc), and is contained in the sendmail implementation of SMTP. This program is provided with most of the UNIX systems today, and has atrocious security history [CHES94]. sendmail contains several tens of thousands of lines of C code, mostly of the spaghetti variety, and runs as *root* (UNIX for "with highest authority"). This is one of the programs exploited by the infamous Internet Worm.

This sendmail bug has been described extensively in the literature, and during the Worm period (November 2-5, 1988), several bug fixes for it were released, mostly by Keith Bostic of the University of California at Berkeley. If any system administrators still run pre-worm versions of sendmail, don't feel sorry for them, they really deserve to have their systems broken into as punishment for being ignorant. Nevertheless, since this is the single most famous bug in the TCP/IP protocol suite, we will take a closer look at it:

The bug consists of a feature that was left in from development, namely the ability to send the sendmail daemon into debug mode. In this mode, the daemon accepts UNIX command line commands, rather than SMTP commands. These command line commands, which are executed at the same authorization level as sendmail, can have very profound effects. The attack involves telnetting to a machine, using the port that the machine normally uses to listen for an SMTP connection. The attacker then types debug on the command line. This causes the daemon to go into debug mode. The attacker can now send commands over to the victims machine in place of normal communication. Two commands which are commonly used by attackers are:

|sed -e '1,\^\\$/d | /bin/sh ; exit 0" - Which strips off the mail header, and executes the rest of the message with root privilege. This command sequence is often used in conjunction with a message body such as: mail attacker@evil.gov </etc/passwd, which will mail the password file to the attacker. The second common command sequence is:

rm -rf /& - which, for those who do not speak UNIX has a result similar to the DOS command sequence: format c: [CHES92] [VENE92]

Other security problems with SMTP include the Multipurpose Internet Mail Extensions (MIME). The MIME system is intelligent in the sense that it can automatically retrieve files from a server for the user. However, the MIME system could just as easily retrieve a file that should not be

retrieved, such as a new .rhosts file. A MIME system that blindly replaces the current version is very dangerous indeed. [CHES94]

Telnet

Telnet is the service on a UNIX system which allows a user to remotely log in to a different machine. It is arguably one of the most useful services on the Internet. For example, if a user who has an account at the University of Washington, goes to Miami to participate in a conference, that user does not have to pay for a long distance call to check her mail. She would simply find a local access number, dial that number, and then telnet whichever account she wanted to use. However, this process is dangerous. Since IP does not have an encryption mechanism (yet. Please see the IPv6 section) the username and password have to travel in plaintext over the network. An attacker grabbing packets could easily figure out passwords this way. There are several ways to get around this, including using one-time passwords, and encrypting telnet packets. Telnet can also be used to mount other attacks, most notably the sendmail attack described above.

Finger

Finger is an extremely useful little program, which every user of the Internet seems to know how to use. Finger will provide the user with information on another user, such as that person's username, home directory, office and telephone, and a plan (usually something like: To graduate from this place this century). All of this is information that a cracker can make great use of. Imagine for example the situation where a cracker is looking for accounts to crack. He could finger any user on host fullerton.edu called Smith. The fingerd on fullerton.edu would then present that cracker with the information on all those users. The cracker could then launch a dictionary attack on the passwords of all those users. Wietse Venema, of Eindhoven University of Technology, stated that after a few days of cracking, 259 out of 1594 passwords were obtained from a set of /etc/passwd files. The Internet worm is estimated to have had a success rate upwards of 50% in some cases [SPAF91]. While this lesson teaches administrators to watch their /etc/passwd files, it also teaches the lesson that, while the finger command is useful, it often provides too much information. Coupled with very weak passwords, it could be extremely dangerous. Another very famous bug in fingerd is the stack problem. This should also be corrected on most systems by now, but again, it is very famous, and therefore deserves mentioning. The fingerd program used a standard routine in C called gets(). This routine does not do any checking that there is enough memory allocated to it. Thus, by causing it to write more information to memory than it is technically allowed to, the behavior of the program can be altered. E.g. by overflowing the stack memory in fingerd, the overflow will be executed as root. gets() is by no means the only C routine that does not do bounds checking. However, it proved to be a crucial problem in the Worm incident [SPAF88]. Patches using a different routine were quickly distributed, and there should not be any old versions of fingerd still in use [EICH89]. There are many other ways to obtain information on users, other than finger. One such service is rpcinfo, which will be discussed next.

RPC - NIS

The Remote Procedure Call protocol (RPC) was developed by Sun Microsystems in order to make network programming slightly easier. RPC allows for replacement of many of the TCP/IP tools, with easier to use RPC tools. RPC also supports the Data Encryption Standard (DES) in the secure RPC implementation. RPC is, however, not immune to many of the TCP attacks, such as the IP spoofing attack, where an attacker purports to be someone else. The authentication in RPC is based on source addresses, and is thus really not worth much, since forging addresses is trivial (as we saw in the SMTP discussion). There are many services running on top of RPC that a cracker is likely to use. For example *rpcinfo*, can tell the cracker many things about your system, such as what file system it is running, whether it is a Network Information Services (NIS) host or server, which processes are running on it etc [FARM93]. This can be very useful for an attacker. For example, NIS is used to distribute information from servers to clients. That in and of itself does not raise many concerns, until one sees what type of information is transmitted: Password files, host address tables, and public and private key databases used for secure RPC. If an attacker can obtain this type of information, your systems processor ticks are probably counted.

File Transfer Protocols

File transfer protocols enable users to transfer files between different computers. Most Internet novices are familiar with the FTP (File Transfer Protocol), but there are two other worth mentioning. The first is the Trivial File Transfer Protocol (TFTP), and the second is FSP (which does not stand for anything).

FTP

FTP sets up a connection between two machines using two TCP connections one command connection, and one data connection. FTP suffers from many of the ordinary security problems inherent in other programs. For example, like the sendmail daemon, ftpd runs as root. Also, just like with telnet, plain-text passwords are passed over unsecure links. However, the most interesting security problem in FTP lies in its common usage as an anonymous service. A recurring problem with FTP-sites is that they have had directories that are both readable and writable to people accessing anonymously. These directories have often been turned into repositories for pornographic material, or pirated software. Another problem is that many utilities running on FTP servers are dependent on the existence of an accessible `/etc/passwd` file. Many system administrators take one of two avenues to solving this problem. The first one is to put a copy of their `/etc/passwd` file in the directory, which is really bad. The second one is to include `/etc` as a directory in the FTP area, which is decidedly worse. Remember, if you were to give a hacker a very appreciated gift, send him your `/etc/passwd` file. An important point to take home here is to not include anything in the FTP area, that is not absolutely necessary there. A number of sites also have so-called drop-directories in that area. These are directories which are

writable but not readable. They are emptied of their contents periodically, and appropriate material is then posted in the appropriate places in the archive.

TFTP

TFTP stands for Trivial File Transfer Protocol. It is commonly used to boot diskless workstations. Unlike FTP, TFTP runs on top of the UDP protocol. The interesting thing is that older TFTP implementations had no restrictions on the files that could be transferred. A cracker could thus TFTP into your system, download the `/etc/passwd` file, and go to work on your network. Sun Microsystems OS prior to release 4.0, for example, did not restrict TFTP. Another really fruitful attack using TFTP would be to use it to put a new `.rhosts` file in a users home directory. As will be explained momentarily, appropriate entries in such a file would allow a cracker to log into your account without even supplying a password [GARF94]. Most experts in the area of Internet security recommend that TFTP not be run at all.

FSP

There is a third file transfer protocol. However, it is so obscure that it is not even mentioned in [GARF94]. It is the FSP, which does not stand for anything [CHES94]. It works similar to FTP but over a UDP connection. Historically it has seldom been used for anything other than bad purposes. Therefore, Cheswick and Bellovin warn administrators that if FSP traffic is being discovered, it is probably bad.

The Berkeley Remote Services

The Berkeley remote services, more often known as the "r" commands were designed to allow users and administrators to work on remote machines as if they were local. There are three criteria for these services:

- 1 The call must originate from a privileged TCP port (usually those with a number below 1024). However, on systems without a concept of ports, such as PCs this restriction cannot be enforced.
- 2 The calling machine must be listed in either the `/etc/hosts.equiv` or the `$HOME/.rhosts` file. (This is why so much emphasis was put earlier on not letting people put these files on other machines).
- 3 The caller's name must correspond to its IP address.

The practical use for these services are that users who use a lot of different machines can switch between them without having to supply a password. Apparently the thought of having to type a password is repugnant to many users. The option of allowing a user to create a `.rhosts` file raises some important security concerns. Is it really prudent to let the users set the security policy for the organization? In one survey, conducted on over 200 hosts, with over 40,000 accounts, close to 10% of the accounts had `.rhosts` files. There were an average of 6 hosts in each. One had over 500 entries! It is hard to conceive of a situation where there are 500 different hosts that need to be authorized to login to an account without passwords [FARM93].

rlogin

rlogin is the r service which allows a user to remotely login to an account without supplying a password. This service is very similar in performance to telnet, other than the lack of password authentication.

rsh

This program lets a user mount a shell on a machine that trusts the user. The remote user can execute a series of commands on the remote machine without actually being connected to it.

rexec

This program works similar to the rsh program, with the exception that it does not present a nice command interpreter as an interface. Basically it was designed to let a system administrator send commands to a remote machine without actually having to log into that machine.

The services, and holes, covered above are only a few of the commonly used holes in the TCP/IP suite. There are innumerable others that a security conscious system administrator needs to be familiar with. Many people simply assume that all the data they have on machines connected to the Internet is public. While this is not a very optimistic outlook on the world, it is probably realistic. There are ways to protect your site, however. Some involve installing all security related updates to system software as soon as they are released. Another involves installing a firewall, essentially a dedicated machine that filters Internet traffic to enforce security policy. While this is not cheap, it is an option that is very often resorted to by many organizations. Beginning with the new version of the Internet Protocol, some of the services provided in these add-on security options will be provided in the network protocol. These services will be discussed next.

IP “the Next Generation”

IPng was the unofficial name given to the newest revision of the Internet protocol, obviously by someone who had watched too much Star Trek. The protocol has now officially been named IPv6, however, the old moniker seems to stick. Much of the information contained in this section of the paper comes from Request For Comment (RFC) number 1752, the *Recommendation for the IP Next Generation Protocol*. This RFC was issued by the IPng Area Directors (IPAD) and was accepted by the Internet Engineering Steering Group (IESG). The specific recommendations of interest from a security standpoint include:

- Support for an authentication header be required
- Support for a specific authentication algorithm be required
- Support for the Privacy Header be required

- Support for a specific privacy algorithm be required
- An IPng framework for firewalls be developed.

IPv6 Addressing Scheme

The main reason that a new version of IP is considered is that the address space in the current version is running out. As stated before, the current addressing scheme consists of a 4-byte address. The new standard proposes to expand this addressing scheme to a 16-byte addressing scheme. This would allow for approximately 3.403×10^{38} hosts. This should for all practical purposes be sufficient for a long time to come. Despite the fact that the address space is four times as long as the IPv4 addresses, the IPv6 header is only twice as long as the IPv4 header. This supposedly maintains efficiency. A very crucial feature of IPv6, though is the ability to append more headers, specifying various options, to the packet. The headers of interest from a security standpoint are the hop limit option in the IP header, the authentication header, and the privacy header.

Hop Limit Option

This option in the standard IPv6 header allows the sender to specify a maximum number of hops (routers traversed) that the packet is allowed to take before it is discarded. This is of some usefulness from a security standpoint. Suppose, for example that an attacker has taken over a routing table that your packet is sent over, and diverted the packets to his account. If that account is farther away, in terms of hops, than the account that the packet was really intended for, it will be discarded before it reaches the attacker. If a connection oriented protocol is used, the sender will be notified of the loss of the packet, and can then take appropriate measures.

Authentication Header

The authentication header is similar to the security label in IPv4. It allows a user to specify the security level of the packet. However, it also includes an additional feature: Authentication Data. This is an algorithm specific piece of information required to authenticate the source of the packet and assure its integrity. This could conceivably be used for such measures as public/private key systems, or digital signatures. This authentication header is a great addition to the very meager security features available in IPv4. It will now be possible to use other methods than source address authentication to authenticate users.

Privacy Header

The privacy header is even more important than the authentication header. The privacy header allows for the encryption of data at the IP level. This header as well starts off with a Security Association Identifier (SAID), which tells the receiver the security level of the data. However, it

also carries a data field, in which encrypted data can be carried. An entire IPv6 datagram can be carried in this field. The header also provides two additional fields, which can prove very useful for various forms of encryption: The initialization vector, and the trailer. The initialization vector can carry synchronization data for a block oriented encryption algorithm. The trailer can carry padding necessary for a block oriented algorithm, or to provide authentication data for algorithms that provide confidentiality without authentication.

IPv6 and Firewalls

The IPv6 also states that an "IPv6 framework for firewalls" be developed. This framework should include information on how a firewall can use the IPv6 authentication header, as well as detail on how IPv6 packets should be analyzed by a firewall. The concern is that many of the firewall configurations in use today would not recognize IPv6 packets, and would thus most likely discard them. Unfortunately, there is, as of yet, precious little information on how this framework for firewalls will be constructed, and what information it will contain.

We now conclude the study of the Internet Protocol and its related protocols security features, and turn our attention to a case study of Internet security. The case selected is the famous Internet Worm Incident, which infected a significant number of machines, on a then significantly smaller Internet, in November of 1988.

Case Study - The Internet Worm

The Internet Worm (or virus, as some authors prefer to call it) was released on the Internet on November 2, 1988. The virus used a number of the features described above to infect machines across the entire Internet. A common estimate of the number of machines infected was that 10%, or 6,000 machines, fell prey to the worm. However, this estimate possesses very little, if any at all, scientific merit. This is based on a guesstimate given by James D. Bruce, MIT EECS Professor and Vice President for Information Systems, and Jeff Schiller, of the Telecommunications Network Group, when pressured by the media as to how many machines were infected. The guesstimate did not intend to represent the number of hosts on the Internet, which were infected, but rather the number of hosts infected at MIT. Gene Spafford, in [SPAF91] gives an estimate of 5% of the machines on the net infected. Regardless of the actual number, the worm prompted many, much-needed modifications to the basic TCP/IP security features, and the author should at least have praise for forcing those to happen. In this case study, we will discuss three main things: What the worm did, how it did it, and what we (hopefully) learned from the incident. A reader who is interested in a further discussion of the worm is referred to [SPAF91], [EICH89] or any of the other numerous papers written on the worm.

What Did the Worm Do?

The most notable effect of the worm was that it used up significant processing time on the affected hosts. This effectively constituted a denial of service attack. Some hosts, such as the gateway machine at the University of Utah reported loads ten times higher than normal, due to multiple infections. The worm also only attacked two types of machines: SUNs and VAXes. These are, however, the single most common machines on the Internet, and the impact was thus severe. The worm basically cracked accounts, in ways to be discussed later, and then launched new attacks from there. In doing so it also spawned new processes (i.e. multiplied) thus spreading rapidly.

More important than what the worm actually did, is what it did not do. For example, the worm did not destroy, nor even attempt to destroy, any data on the host machine. If the author had wanted to, he could have easily destroyed most of the data on the infected machines. However, no attempt to do so was made. The worm also did not normally attempt to gain privileged access. It almost never broke into a system as root. In these two ways it definitely differs from normal cracker attacks, which often have destruction as their purpose. The worm also did not leave any timebombs behind. Most viruses and worms on PCs leave processes to be executed at a specific time, ranging from an annoying message or a song, to re-formatting of a hard-drive. The author of the Internet worm never attempted to leave any such time-bombs behind.

How Did the Worm Operate?

This really boils down to what security holes the author utilized. There were four major features exploited by the worm. These features contributed to its rapid spread.

The first feature of the worm exploited the sendmail bug, described above, in the following manner: It initiated an SMTP connection to a remote site by simply telnetting to the port that SMTP normally uses for connections. The worm then sent the command debug to the daemon. This sent sendmail into debug mode. The worm sent a program over, in the recipient field. This program (which was a shell program) created a C program. This C program, in turn contacted the attacking machine and downloaded a set of C object files. These files, which contained the actual worm program, were linked and executed, thereby infecting another machine.

If the sendmail attack was unsuccessful, the worm could try to spawn a remote shell by invoking the rsh service. This shell would then use the same infection steps as in the discussion of the sendmail infection above.

The third way that the worm attacked was by using the bug described in the finger section above. Basically, it involved rewriting a portion of the stack used by fingerd, to execute a command line, which allowed the worm to connect to a remote shell. Not all of these break in attempts were used. As soon as one succeeded, the worm started cracking passwords.

Once the worm had achieved entry via one of the above features, it proceeded to utilize the information on the infected host to infect other systems. It read the systems `/etc/hosts.equiv` file. This file lists other systems which the host trusts. The worm used this file to find machine names that would be likely targets. It also read the `$HOME/.rhosts` file. This file provides user-specific information of the same form as the `/etc/hosts.equiv` file. In addition the program read the entire `/etc/passwd` file of the infected system. It then used both a built-in dictionary, and the system's own dictionary to launch a dictionary attack against all the accounts listed in that file. Once it cracked passwords in this file, it searched the cracked accounts for personal `.forward` files (these files are used to provide the SMTP daemon with systems to which it should autoforward mail), in search of other machines to attack. Once it had found a password, it also attempted to use it to connect to accounts given in the `.rhosts`, and `.forward` files. There were many more features of the worm, some of which checked if other worms ran on a newly infected machine, and others which changed process ID numbers. However, for the purposes of our discussion here, the above features show how dangerous the holes discussed in this paper can be. It is more important to discuss what we have learned from the worm.

Lessons To Be Learned

The first lesson to be learned is probably that connectivity really saved the network. One can argue that connectivity was what allowed the worm to spread. However, connectivity also allowed people in the know throughout the net to communicate, post bug fixes, and crack the worm. In addition, a very important lesson was that bug fixes are critical. These fixes sometimes were as simple as renaming a couple of files on the computer. Sometimes, they involved a binary edit of a daemon. In any case, they were very important. The worm incident also showed how important it is to have adequate security policies. It is absolutely unacceptable to have 50% of your passwords broken by a relatively simple dictionary attack. There are features available to enforce good password selection, and frequent change, and these should be installed on any system. Also, it is up to each system administrator to decide how much security policy is put in the users' hands. By allowing the Berkeley "r" services on a system, the administration of security policy is effectively transferred from the administrator to the user.

Conclusion

This paper has presented several security issues related to the TCP/IP protocol suite. The main lesson to be learned is that data on unprotected computers is apt to be read by anyone who wishes to do so. We have also discussed several new features which will be present in the next generation IP protocols. The next generation of the Internet protocol will certainly make the task of managing security easier. However, since many of the problems discussed are contained in the higher level protocols, IPv6 will not resolve those. By no means should this be construed as an exhaustive discussion of TCP/IP security problems and features. Rather, it should be considered a preliminary primer on some of the issues that the security conscious manager,

implementing Internet connections for a business, need to concern him/herself with. There is only one foolproof way to protect your system. It involves disconnecting all network cables, putting the computer inside a vault, and post a 24-hour guard outside. For those who will not consider such measures, the Internet presents active opportunities, both for contact with customers, and for intra-company communication. However, the manager considering hooking a system up to the Internet need to seriously consider the security aspects of doing so. Hopefully, this paper has provided a few insights on what to look at.

AINT MISBEHAVING -- A TAXONOMY OF ANTI-INTRUSION TECHNIQUES

Lawrence R. Halme
(halme@arca.ca.com)
R. Kenneth Bauer
(bauer@arca.ca.com)
Arca Systems, Inc.†
2540 North First St., Suite 301
San Jose, CA 95131-1016

Abstract: This paper examines the basic underlying principles of intrusion control and distills the universe of anti-intrusion techniques into six high-level, mutually supportive approaches. System and network intrusions may be prevented, preempted, deflected, deterred, detected, and/or autonomously countered. This Anti-Intrusion Taxonomy (AINT) of anti-intrusion techniques considers less explored approaches on the periphery of "intrusion detection" which are independent of the availability of a rich audit trail, as well as better known intrusion detection techniques. Much like the Open Systems Reference Model supports understanding of communications protocols by identifying their layer and purpose, the authors believe this anti-intrusion taxonomy and associated methods and techniques help clarify the relationship between anti-intrusion techniques described in the literature and those implemented by commercially available products. The taxonomy may be used to assess computing environments which perhaps already support Intrusion Detection System (IDS) implementations to help identify useful complementary intrusion defense approaches.

Keywords: Intrusion, detection, misuse, anomaly, countermeasure, taxonomy.

1.0 Introduction

Efforts to combat computer system intrusions have historically included preventive design, configuration, and operation techniques to make intrusion difficult. Acknowledging that by bowing to functionality concerns and budgetary constraints these efforts will be imperfect, the concept was suggested to detect intru-

sions by analyzing collected audit data. The study of anomaly detection was prefaced by the postulate that it would be possible to distinguish between a masquerader and a legitimate user by identifying deviation from historical system usage [AND80]. It was hoped that an audit analysis approach would be useful to identify not only crackers who had acquired identification and authentication information to permit masquerading as legitimate users, but also legitimate users who were performing unauthorized activity (misfeasors). Clandestine users able to bypass the security mechanisms were another identified problem, but considered more difficult to detect since they could influence system auditing.

Early hands-on experimentation confirmed that user work patterns could be distinguished using existing audit trails [HAL86]. Techniques were debated to make auditing, which was originally designed primarily for accounting purposes, more useful to security analysis. A model was developed which theorized much of the framework for a general-purpose intrusion detection system [DEN87]. Intrusion detection researchers split into two camps — those seeking attack signatures in the audit data which announce known misuse (e.g., MIDAS [SEB88]), and those seeking evidence of usage which is anomalous from historical norms (e.g., IDES [LUN88a]). The complementary combination of these approaches into an investigative tool with autonomous response to particularly threatening deviance was suggested [HAL88]. Survey papers attest to the dramatic growth in the number of research efforts investigating different anomaly and misuse detection approaches ([LUN88b], [TIS90]).

†This work was sponsored by the Air Force Information Warfare Center.

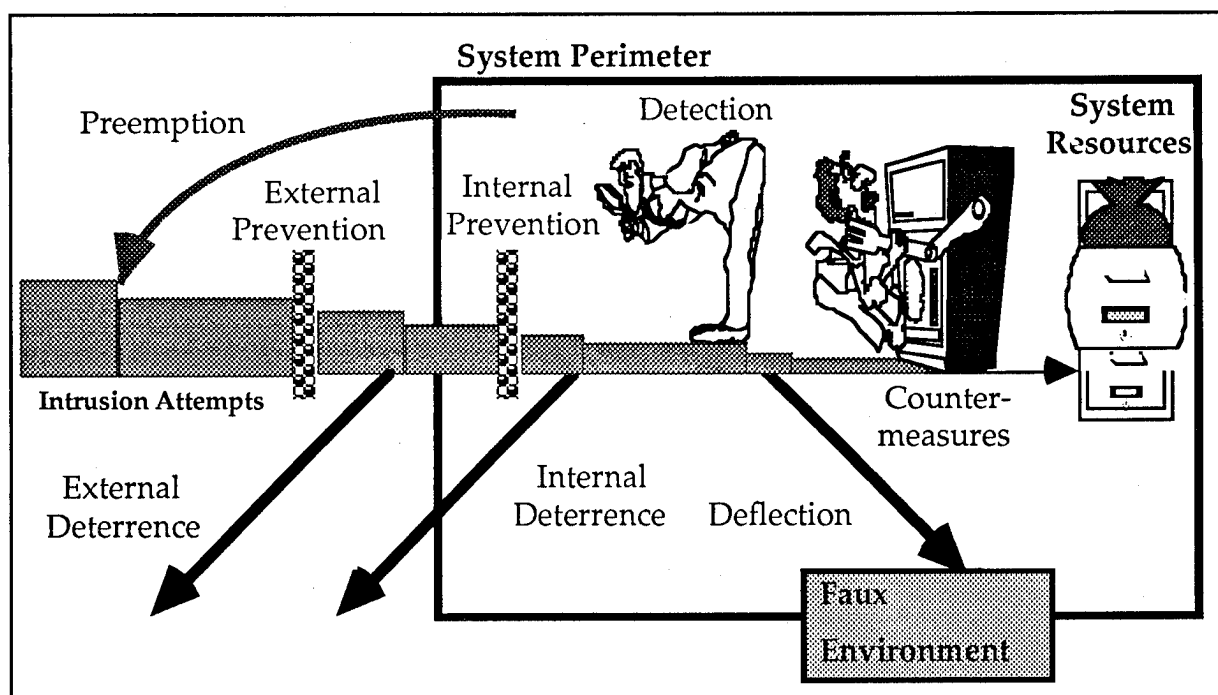
The early Nineties saw test and commercial installation and operation of a number of IDS's including SRI's IDES and NIDES, Haystack Laboratory Inc.'s Haystack and Stalker, and the Air Force's Distributed Intrusion Detection System (DIDS). Emphasis broadened to include integration of audit sources from multiple heterogeneous platforms, and platform portability. Distributed intrusion detection is the focus of work at the University of California at Davis [HEBE92] and at the Air Force [DIDS91]. Intrusion detection continues to be an active field of research.

Although much has been learned from these research-driven efforts, their focus has been on developing optimized techniques to detect intrusions. Less thought has been given to creating an operational view of complementary anti-intrusion approaches. Computer and Internet misuse has become a frequent topic of today's mainstream media, and the demand for anti-

motes multiple approach solutions.

2.0 Anti-Intrusion Approaches

Over the past fifteen years a great deal of emphasis has been placed on detection as the most fruitful area for research and development to combat intrusion activity (both from external crackers as well as insiders abusing their privileges). Less considered have been other complementary anti-intrusion techniques which can play valuable roles. As work environments become more interconnected and exposed, service providers will need increasingly to rely on a wide range of anti-intrusion techniques, not just IDS's. This paper organizes these techniques (illustrated in Figure 1) into the Anti-Intrusion Taxonomy (AINT). The "filtering" of successful intrusions is graphically depicted by the narrowing of the successful intrusion attempt band.



intrusion technology is exploding. However, intrusion detection products are as yet esoteric and not well integrated to work together with complementary approaches such as intrusion preventing firewalls. The taxonomy we present in this paper seeks to give perspective and aid understanding. It provides the basis for the formulation of a systematic and comprehensive anti-intrusion approach categorization and pro-

Figure 1: Anti-Intrusion Approaches

The following text describes the six anti-intrusion approaches. We also provide an analogous real-world illustration of each approach as applied to combating the possibility of having your wallet stolen walking down an urban street. Sections follow which elaborate how these approaches apply to computer systems under the AINT.

Prevention precludes or severely handicaps the likelihood of a particular intrusion's success.

Hire hulking bodyguards and avoid bad neighborhoods. A definitive approach when it works, but expensive and troublesome and unlikely to be operationally 100% foolproof. Still leaves opportunity for successful attack if bodyguards can be distracted or bribed.

Preemption strikes offensively against likely threat agents prior to an intrusion attempt to lessen the likelihood of a particular intrusion occurring later.

Support vigilante patrols. Non-specific and may affect innocents.

Deterrence deters the initiation or continuation of an intrusion attempt by increasing the necessary effort for an attack to succeed, increasing the risk associated with the attack, and/or devaluing the perceived gain that would come with success.

Dress down and walk with excitable Chihuahua dog. Many attackers will move on to richer looking easier prey, but if it has been a lean night, a little annoying yapping dog isn't going to stop a determined mugger.

Deflection leads an intruder to believe that he has succeeded in an intrusion attempt, whereas instead he has been attracted or shunted off to where harm is minimized.

Carry two wallets so that when attacked, a decoy wallet with canceled credit cards can be handed over. Can learn more about how attackers operate, but probably only works for newbie muggers and it is inconvenient having to carry two wallets.

Detection discriminates intrusion attempts and intrusion preparation from normal activity and alerts the authorities.

Carry a whistle and blow to attract attention from beat cop if attacked. Limited usefulness if attack is too far from a donut shop for whistle to be heard, or if car-alarm-syndrome causes authorities to ignore as a false alarm. Also you may not detect in time that your wallet was stolen if it is surreptitiously pickpocketed.

Countermeasures actively and autonomously counter an intrusion as it is being attempted.

Carry a can of mace, attach mouse trap to wallet, and know karate to counter attack. Run the risk of being

sued by accidentally breaking the arm of Hari Krishna solicitor offering flowers. With a booby trapped wallet, a pickpocket can be autonomously countered with necessary speed without conscious detection. However you, as an authorized user, might mistakenly get your fingers snapped if you forget about the mouse-trap.

3.0 Intrusion Prevention

Intrusion Prevention techniques (enforced internally or externally to the system) seek to preclude or at least severely handicap the likelihood of success of a particular intrusion. These techniques help ensure that a system is so well conceived, designed, implemented, configured, and operated that the opportunity for intrusions is minimal. Because built-in prevention seeks to make it impossible for an intrusion to occur on the target system, it may be considered the strongest anti-intrusion technique. Ideally, this approach would prevent all intrusions, negating the need for detection and consequent reaction techniques. Nevertheless, in a real world system this technique alone proves untenable and unlikely to be implemented without some remaining exploitable faults and dependence on configuration/maintenance. Add-on prevention measures augmenting the defenses of an existing system include vulnerability scanning tools and network firewalls.

Correct Design / Implementation techniques represent classic INFOSEC mechanisms (e.g., identification and authentication, mandatory and discretionary access control, physical security), and are appropriate to be developed into the target system itself. These techniques are well explored, but may be cumbersome and expensive, and care must be taken that they are not poorly configured.

Vulnerability Scanning Tools examine system and network configurations for oversights and vulnerabilities. Static configuration scanners are programs and scripts periodically run manually by the System Security Officer (SSO) to detect system vulnerabilities. Dynamic configuration scanning tools perform much the same function but run constantly as a low priority task in the background. Configuration scanning tools can monitor for a wide range of system irregularities including: unauthorized software, unauthorized accounts, unprotected logins, inappropriate resource

ownership, inappropriate access permissions, weak passwords, and ghost nodes on a network. Other vulnerability scanning tools can check for evidence of previous intruder activity, susceptibility to known attacks, and dormant viruses. Representative UNIX configuration scanning tools include: Security Profile Inspector (SPI), Internet Security Scanner (ISS), Security Analysis Tool for Auditing Networks (SATAN), COPS, and Tripwire [FIS94].

Firewalls examine and control the flow of information and services between a protected subnetwork and/or hosts and the outside world. They protect one network from another by blocking specific traffic while allowing other traffic. The most common use today is connecting corporate and academic networks to the Internet. Firewall designs have proven effective in thwarting many intruder efforts. The decision as to which traffic to allow is based upon the content of the traffic itself. Typical decision criteria include traffic direction, network address, port, protocol type, and service type. The goal of the firewall is to provide efficient and authorized access for users "inside" the firewall to the outside world while controlling the access of "outside" users to protected resources by exporting limited and precisely controlled services. Firewalls are best implemented on separate hardware for performance and security reasons, and thus there is expense of acquisition and maintenance.

4.0 Intrusion Preemption

Intrusion Preemption techniques strike offensively prior to an intrusion attempt to lessen the likelihood of a particular intrusion occurring later. This approach includes such techniques as education of users, promoting legislation to help eliminate an environment conducive to intrusion, taking early action against a user who appears increasingly to be straying from the straight-and-narrow, and infiltrating the cracker community to learn more about techniques and motivation. Rather than the reactive defenses offered by detection and countermeasures, preemption refers to proactive action against the source of as yet unlaunched intrusions. Unchecked use of these techniques can pose civil liberty questions.

Banishment refers to producing a hostile environment intended to reduce the ranks of potential intruders prior to their attempt to launch an intrusion. Users

can be educated about security threats from technical and nontechnical attacks, and provided directives on how to handle specific social engineering information requests. Support of legislation which deals harshly with intruders is another example of this technique.

Vigilance seeks to preempt later intrusions by noticing preliminary danger signs of impending undesired activity. Examples of this technique include attempting to discern malicious intent and initial exploratory stages of intrusion activity, taking strong and early action against users demonstrating a leaning toward violating system policy, and offering to reward users who spot vulnerabilities or unauthorized usage.

Infiltration refers to proactive efforts on the part of the SSO to acquire attack information from underground sources to supplement vendor bug reports and Computer Emergency Response Team (CERT) warnings. A more insidious infiltration would inundate hacker bulletin board systems with false information to confuse and discourage.

5.0 Intrusion Deterrence

Intrusion Deterrence seeks to make any likely reward from an intrusion attempt appear more troublesome than it is worth. Deterrents encourage an attacker to move on to another system with a more promising cost-benefit outlook. This approach includes devaluing the apparent system worth through camouflage, and raising the perceived risk of being caught by displaying warnings, heightening paranoia of active monitoring, and establishing obstacles against undesired usage. Intrusion deterrents differ from intrusion prevention mechanisms in that they are weaker reminder/discomfort mechanisms rather than serious attempts to preclude an intrusion.

Camouflage seeks to hide and/or devalue system targets and encompasses such straightforward policy as minimizing advertising a system and its contents. Configuring a dial-in line not to pick up for a number of rings greater than most cracker demon dialing software, and presenting only generic logic banners are other examples of camouflage. A faceless, boring system is not a prize trophy for a cracker. A disk entitled "Thermonuclear War" intrigues more than one deglamourized to "tnw." Camouflage may make a

system less usable and intuitive. It also may conflict with the following deterrent techniques which seek to emphasize active defenses. However, a system that reveals efforts to secure it may beg an attacker to investigate why such effort was expended. Simple and weak camouflage techniques may nonetheless prove useful as deterrents to intrusion.

Warnings inform users that the security of a system is taken seriously and emphasizing what the penalties are if unauthorized activity is monitored. Sensitive systems are often configured to display warnings as part of their standard login banners. Users not contemplating an intrusion should be little inconvenienced. Warnings are easily implemented and may also be useful from a legal standpoint (especially in the case of keystroke monitoring), but if the intruder perceives all-bark-no-bite, this is a weak defense. Warnings may even be counterproductive by piquing the curious, and laying down a provocative gauntlet to intruders out to prove their mettle. Particular user warnings may also be implemented to trigger when specific undesirable activity is detected. A concern for activity-based user warnings is that the potential intruder is alerted to what thresholds/signatures fire the anti-intrusion mechanism.

Paranoia refers to increasing the impression (whether true, exaggerated, or fallacious) that user activity is being closely monitored by a vigilant SSO. Where having nonstop watchful system administration in place is not practical, it may be simulated. If the intruder is led to believe the risks of detection and prosecution from an apparently attentive and motivated SSO are greater than the possible reward, he may instead move on to "easier pickings." Emulating the "fake car alarm blinking light" mechanism is the simplest technique to give the misleading impression of constant live monitoring. A "scarecrow" process performing semi-random standard system administrator activities may be sufficient to ward off casual intruders who have not seriously cased the system. The deterrent value of this technique is lost, however, as soon as potential intruders learn that a Scarecrow is present and learn ways to distinguish between the Scarecrow and a real SSO. An enhancement to this is to implement a "security camera" technique which admittedly only randomly offers live-monitoring, but gives no indication when the SSO is actually watch-

ing. A potential intruder in this case can never be sure when he is actually being live-monitored, but is aware that it may be at any time.

Obstacles seek to increase the ante of time and effort an attacker must expend to succeed beyond what the perceived reward warrants. Obstacles, especially on gateway machines, seek to try the patience of an intruder thereby "ruining his fun" and providing incentive to move on. Delaying command executions, displaying false system warnings, apparent exhaustion of resources, and similar obstacles serve to exasperate, but not advertise detection. Annoying tactics may include showing interesting but dead-end lures — dummy accounts or files on which the intruder wastes valuable time and reveals attack skills, but which award him nothing. Use of this technique risks inconveniencing authorized users.

6.0 Intrusion Deflection

Intrusion Deflection dupes an intruder into believing that he has succeeded in accessing system resources, whereas instead he has been attracted or shunted to a specially prepared, controlled environment for observation (i.e., a "playpen" or "jail"). Controlled monitoring of an unaware intruder spreading out his bag of tricks is an excellent source of attack information without undue risk to the "real" system [ST089]. Some system enforced deflection techniques may be considered a special type of countermeasure, but the concept also includes techniques which do not require the protected system to have ever been accessed by the intruder (e.g., "lightening-rod systems").

Quarantined Faux Systems are designed to lead intruders (primarily the unfamiliar "outsider") to believe that they are logged into the target system, when they are actually locked into a separate "fish-bowl" system. This deflection is accomplished by a network front end system such as a router or firewall. An effective quarantined faux system encourages an intruder to remain long enough for a response team to determine the intruder's identity and motive. However, dedicating a separate machine and the resources to maintain this charade is expensive, and with distributed environments and the powerful statusing tools available, this technique may be untenable.

Controlled Faux Accounts are designed to lead intruders to believe that they are executing within a

compromised standard account, when instead they are locked into a special limited access account. In this case, the deflection controls are built right into the target environment operating system or application. This technique eliminates the need for the separate hardware resources required by a faux system, but must rely on the target operating system security to ensure isolation from protected system resources. The constructed environment could contain various inducements to engage and stall the intruder, and divulge his intent. However, constructing and maintaining a believable and unbreakable controlled faux account is difficult.

Lightning Rod Systems / Accounts are similar to the preceding faux techniques, but rather than the intruder being unknowingly shunted to them, the intruder is instead lured into pursuing a decoy controlled environment directly of his own volition. Lightning rod systems are placed "near" assets requiring protection, are made attractive, and are fully instrumented for intrusion detection and back tracking (the term "honey pot" has also been used to describe this technique). They are distinct from the primary resources being protected, and do not need to be concerned about performance and functionality handicaps to authorized users. A practical and convincing implementation of nontrivial lightning rods is problematic: they are likely expensive to install and maintain, and rely upon their true reason for existence remaining secret.

7.0 Intrusion Detection

Intrusion Detection encompasses those techniques that seek to discriminate intrusion attempts from normal system usage and alert the SSO. Typically, system audit data is processed for signatures of known attacks, anomalous behavior, and/or specific outcomes of interest. Intrusion detection, and particularly profiling, is generally predicated upon the ability to access and analyze audit data of sufficient quality and quantity. If detection is accomplished in near real-time, and the SSO is available, he could act to interrupt the intrusion. Because of this necessity for a human to be available to intervene, Intrusion Detection is not as strong an approach as Intrusion Countermeasures as it is more likely that intrusion efforts will complete before manual efforts can interrupt the

attack. Intrusion Detection may be accomplished after the fact (as in postmortem audit analysis), in near-real time (supporting SSO intervention or interaction with the intruder, such as network trace-back to point of origin), or in real time (in support of automated countermeasures).

7.1 Anomaly Detection

Anomaly Detection compares observed activity against expected normal usage profiles which may be developed for users, groups of users, applications, or system resource usage. Audit event records which fall outside the definition of normal behavior are considered anomalies.

Threshold Monitoring sets values for metrics defining acceptable behavior (e.g., fewer than some number of failed logins per time period). Thresholds provide a clear, understandable definition of unacceptable behavior and can utilize other facilities besides system audit logs. Unfortunately it is often difficult to characterize intrusion behavior solely in terms of thresholds corresponding to available audit records. It is difficult to establish proper threshold values and time intervals over which to check. Approximation can result in a high rate of false positives, or high rate of false negatives across a non-uniform user population.

User Work Profiling maintains individual work profiles to which the user is expected to adhere in the future. As the user changes his activities his expected work profile is updated. Some systems attempt the interaction of short-term versus long-term profiles; the former to capture recent changing work patterns, the latter to provide perspective over longer periods of usage. However it remains difficult to profile an irregular and/or dynamic user base. Too broadly defined profiles allow any activity to pass review.

Group Work Profiling assigns users to specific work groups which demonstrate a common work pattern and hence a common profile. A group profile is calculated based upon the historic activities of the entire group. Individual users in the group are expected to adhere to the group profile. This method can greatly reduce the number of profiles needing to be maintained. Also a single user is less able to "broaden" the profile to which they are to conform. There is little

operational experience with choosing appropriate groups (i.e., users with similar job titles may have quite different work habits). Individual user profiles mimicked by creating groups of one may be a necessary complication to address users who do not cleanly fit into the defined groups.

Resource Profiling monitors system-wide use of such resources as accounts, applications, storage media, protocols, communications ports, etc., and develops a historic usage profile. Continued system-wide resource usage—illustrating the user community's use of system resources as a whole—is expected to adhere to the system resources profile. However, it may be difficult to interpret the meaning of changes in overall system usage. Resource profiling is user-independent, potentially allowing detection of collaborating intruders.

Executable Profiling seeks to monitor executables' use of system resources, especially those whose activity cannot always be traced to a particular originating user. Viruses, Trojan horses, worms, trapdoors, logic bombs and other such software attacks are addressed by profiling how system objects such as files and printers are normally used, not only by users, but also by other system subjects on the part of users. In most conventional systems, for example, a virus inherits all of the privileges of the user executing the infected software. The software is not limited by the principle of least privilege to only those privileges needed to properly execute. This openness in the architecture permits viruses to surreptitiously change and infect totally unrelated parts of the system. User-independent executable profiling may also be able to detect collaborating intruders.

Static Work Profiling updates usage profiles only periodically at the behest of the SSO. This prevents users from slowly broadening their profile by phasing in abnormal or deviant activities which are then considered normal and included in the user's adaptive profile calculation. Performing profile updates may be at the granularity of the whole profile base or, preferably, configurable to address individual subjects. SSO controlled updates allow the comparison of discrete user profiles to note differences between user behavior or changes in user behavior. Unfortunately these profiles must either be wide and insensitive or frequently updated. Otherwise if user work

patterns change significantly, many false positives will result — and we all recall the story of Peter and the Wolf. This approach also requires diligence on the part of the SSO who must update profiles in response to false positives, and ensure changes represent legitimate work habit changes.

Adaptive Work Profiling automatically manages work profiles to reflect current (acceptable) activity. The work profile is continuously updated to reflect recent system usage. Profiling may be on user, group, or application. Adaptive work profiling may allow the SSO to specify whether flagged activity is: 1) intrusionary, to be acted upon; 2) not intrusionary, and appropriate as a profile update to reflect this new work pattern, or 3) not intrusionary, but to be ignored as an aberration whose next occurrence will again be of interest. Activity which is not flagged as intrusionary is normally automatically fed into a profile updating mechanism. If this mechanism is automated, the SSO will not be bothered, but work profiles may change and continue to change without the SSO's knowledge or approval.

Adaptive Rule Based Profiling differs from other profiling techniques by capturing the historical usage patterns of a user, group, or application in the form of rules. Transactions describing current behavior are checked against the set of developed rules, and changes from rule-predicted behavior flagged. As opposed to misuse rule-based systems, no prior expert knowledge of security vulnerabilities of the monitored system is required. "Normal usage" rules are generated by the tool in its training period. However, training may be sluggish compared to straight statistical profiling methods. Also, to be effective, a vast number of rules must be maintained with inherent performance issues. Management of tools adopting this technique require extensive training, especially if site-specific rules are to be developed.

7.2 Misuse Detection

Misuse detection essentially checks for "activity that's bad" with comparison to abstracted descriptions of undesired activity. This approach attempts to draft rules describing known undesired usage (based on past penetrations or activity which is theorized would exploit known weaknesses) rather than describing historical "normal" usage. Rules may be written to

recognize a single auditable event that in and of itself represents a threat to system security, or a sequence of events that represent a prolonged penetration scenario. The effectiveness of provided misuse detection rules is dependent upon how knowledgeable the developers (or subsequently SSO's) are about vulnerabilities. Misuse detection may be implemented by developing expert system rules, model based reasoning or state transition analysis systems, or neural nets.

Expert Systems may be used to code misuse signatures as if-then implication rules. Signature analysis focuses on defining specific descriptions and instances of attack-type behavior to flag. Signatures describe an attribute of an attack or class of attacks, and may require the recognition of sequences of events. A misuse information database provides a quick-and-dirty capability to address newly identified attacks prior to overcoming the vulnerability on the target system. Typically, misuse rules tend to be specific to the target machine, and thus not very portable.

Model Based Reasoning attempts to combine models of misuse with evidential reasoning to support conclusions about the occurrence of a misuse. This technique seeks to model intrusions at a higher level of abstraction than the audit records. In this technique, SSO's develop intrusion descriptions at a high, intuitive level of abstraction in terms of sequences of events that define the intrusion. This technique may be useful for identifying intrusions which are closely related, but whose audit trails patterns are different. It permits the selective narrowing of the focus of the relevant data, so a smaller part of the collected data needs to be examined. As a rule-based approach it is still based on being able to define and monitor known intrusions, whereas new and unknown vulnerabilities and attacks are the greatest threats.

State Transition Analysis creates a state transition model of known penetrations. In the Initial State the intruder has some prerequisite access to the system. The intruder executes a series of actions which take the target system through intermediate states and may eventually result in a Compromised State. The model specifies state variables, intruder actions, and defines the meaning of a compromised state. Evidence is preselected from the audit trail to assess the possibility that current system activity matches a modeled sequence of intruder penetration activity (i.e., de-

scribed state transitions lead to a compromised state). Based upon an ongoing set of partial matches, specific audit data may be sought for confirmation. The higher level representation of intrusions allows this technique to recognize variations of scenarios missed by lower level approaches.

Neural Networks offer an alternative means of maintaining a model of expected normal user behavior. They may offer a more efficient, less complex, and better performing model than mean and standard deviation, time decayed models of system and user behavior. Neural network techniques are still in the research stage and their utility have yet to be proven. They may be found to be more efficient and less computationally intensive than conventional rule-based systems. However, a lengthy, careful training phase is required with skilled monitoring.

7.3 Hybrid Misuse / Anomaly Detection

Hybrid Detectors adopt some complementary combination of the misuse and anomaly detection approaches run in parallel or serially. Activity which is flagged as anomalous may not be noticed by a misuse detector monitoring against descriptions of known undesirable activity. For example, simple browsing for files that include the string "nuclear" may not threaten the security or integrity of the system but it would be useful information for an SSO to review if it was anomalous activity for a particular account. Likewise, an administrator account may often demonstrate access to sensitive files and have a profile to permit this, but it would be useful for this access to still be checked against known misuse signatures. There has been a fairly strong consensus in the anti-intrusion community that effective and mature intrusion detection tools need to combine both misuse and anomaly detection. There is increasing operational field evidence that anomaly detection is useful, but requires well briefed SSO's at each site to configure and tune the detector against a high rate of false positives. Anomaly detection systems are not turnkey and require sophisticated support at least until profiles have stabilized.

7.4 Continuous System Health Monitoring

Intrusions may be detected by the continuous active monitoring of key "system health" factors such as performance and an account's use of key system resources. This technique is more flexible and sophisticated than Static Configuration Checkers, as such a tool would be run continuously as a background process. It concentrates on identifying suspicious changes in system-wide activity measures and system resource usage. An example is to monitor network protocol usage over time, looking for ports experiencing unexpected traffic increases. Work needs to be done to develop and tune system-wide measures, and to understand the significance of identified variations.

8.0 Intrusion Countermeasures

Intrusion Countermeasures empower a system with the ability to take autonomous action to react to a perceived intrusion attempt. This approach seeks to address the limitation of intrusion detection mechanisms which must rely on the constant attention of an SSO. Most computing environments do not have the resources to devote an SSO to full-time intrusion detection monitoring, and certainly not for 24 hours a day, seven days a week. Further, a human SSO will not be able to react at machine processing speeds if an attack is automated — the recent IP spoofing attack attributed to Kevin Mitnick was largely automated and completed in less than eight minutes [SHI95]. Entrusted with proper authorization, a system will have much greater likelihood of interrupting an intrusion in progress, but runs the risk of falsely reacting against valid usage. What must be prevented is the case where a user is doing something unusual or suspicious, but for honest reasons, and is wrongfully burdened by a misfiring countermeasure. The concern that a General Brassknuckles will be enraged by being rudely locked out of the system because he runs over the allowed page count for printouts, merely reflects an avoidable, overly aggressive countermeasure configuration.

Two primary intrusion countermeasure techniques are autonomously acting IDS's and alarmed system resources. Although the former may be considered simply giving intrusion detection techniques teeth, the latter will react to suspicious actions on the system without ever processing audit data to perform "detection."

Intrusion Countermeasure Equipment (ICE)¹ refer to mechanisms which not only detect but also autonomously react to intrusions in close to real-time. Such a tool would be entrusted with the ability to take increasingly severe autonomous action if damaging system activity is recognized, especially if no security operator is available. The following ICE autonomous actions, in ascending order of severity, may be envisioned:

Alert, Increase Support to SSO (Transparent):

- Note the variance in ICE console window
- Increase the amount of audit data collection on the irregular user, perhaps down to the keystroke level
- Alert SSO at the ICE console with a local alarm
- Notify SSOs remotely (e.g., by beeper)

Seek to Confirm, Increase Available Information on User:

- Reauthenticate user or remote system (i.e., to address attacks originating from intruders capitalizing on an unattended session, or spoofing packets on an authenticated connection)
- Notify security personnel to get voice/visual confirmation of the user's identity/intention

Minimize Potential Damage:

- Slow system response or add obstacles
- Only pretend to execute commands (e.g., buffer rather than truly delete)

Arrest Continued Access:

- Lock local host account / Swallow offending packets
- Trace back network ID and lock out all associated accounts back to entering host, perform housekeeping at intermediary systems.
- Lock entire host system / Disconnect from network
- Disconnect network from all outside access

ICE offers a number of advantages over manually reviewed IDS's. A system can be protected without requiring an SSO to be constantly present, and able and willing to make instant, on-the-spot complex decisions. ICE offers non-distracted, unbiased, around-the-clock response to even automated attacks. Because ICE suffers from the same discrimination and profile management issues as intrusion detection mechanisms, but with potentially no human intervention, care must be taken that service is not disrupted at a critical time by engineered denial of service attacks.

¹The anti-intrusion term "ICE" originated from science fiction author William Gibson's seminal cyberpunk novel *Neuromancer*, and was appropriated and modified by [HAL88]. Mr. Gibson was reportedly amused by this instance of life mimicking art.

Alarmed Files/Accounts refer to seductively named and strategically located "booby trap" resources which lure an intruder into revealing his activities. Accessing an alarmed file or account unleashes immediate action. Alarms can be silent (only notifying the SSO, even remotely) or can prompt immediate retaliatory action against the intruder. An ideal candidate for an alarmed account is a default administrator account with default password intact. This technique is low cost and low tech, but care must be taken that authorized users will not trip the alarm, especially through accidental stumbling across it by some automatic means (e.g., running a nonmalicious find).

9.0 Conclusion

This paper has established a comprehensive anti-intrusion taxonomy by working top-down at a theoretical level, and bottom-up by surveying implemented approaches and those discussed in the referenced literature. Exercising the taxonomy against real life analogies firmed and increased intuitive grasp of the concepts. New anti-intrusion techniques will continue to be developed in this rapidly evolving field of research which may expand our taxonomy. This taxonomy will serve as a useful tool to catalog and assess the anti-intrusion techniques used by a particular anti-intrusion system implementation. It is hoped that our technique will provide new insight to the anti-intrusion research community. The authors are active workers in the field and would be pleased to correspond regarding additions or modifications.

10.0 References

- [AND80] J.P. Anderson. *Computer Security Threat Monitoring and Surveillance*. James P. Anderson Co., Fort Washington, PA, 15 April 1980.
- [HAL86] L. Halme and J. Van Horne. "Automated Analysis of Computer System Audit Trails for Security Purposes," *Proceedings of the 9th National Computer Security Conference*. Washington DC. September 1986.
- [DEN87] D. Denning. "An Intrusion Detection Model," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2. February 1987. pp. 222-232.
- [SEB88] E. Sebring, E. Shellhouse, M. Hanna, and R. Whitehurst. "Expert Systems in Intrusion Detection: A Case Study," *Proceedings of the 11th National Computer Security Conference*. Washington DC. October 1988.
- [LUN88a] T. Lunt and R. Jagannathan. "A Prototype Real-Time Intrusion Detection Expert System," *Proceedings of the 1987 IEEE Symposium on Security and Privacy*. Oakland CA. April 1988.
- [HAL88] L. Halme and B. Kahn. "Building a Security Monitor with Adaptive User Work Profiles," *Proceedings of the 11th National Computer Security Conference*. Washington DC. October 1988.
- [LUN88b] T. Lunt. "Automated Audit Analysis and Intrusion Detection: A Survey," *Proceedings of the 11th National Computer Security Conference*. Washington DC. October 1988.
- [TIS90] N. McAuliffe, D. Wolcott, L. Schaefer, N. Kelem, B. Hubbard, T. Haley. "Is Your Computer Being Misused? A Survey of Current Intrusion Detection System Technology," *Proceedings of the 6th Annual Computer Security Applications Conference*. Tucson, AZ. December 1990.
- [HEBE92] L. Heberlein, B. Mukherjee, K. Levitt. "Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks," *Proceedings of the 15th National Computer Security Conference*. Washington DC. October 1992.
- [DIDS91] S. Snapp, J. Brentano, G. Dias, T. Goan, L. Heberlein, C. Ho, K. Levitt, B. Mukherjee, S. Smaha, T. Grance, D. Teal, and D. Mansur. "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and an Early Prototype," *Proceedings of the 14th National Computer Security Conference*. October 1991.
- [FIS94] W. Cheswick and S. Bellovin. *Firewalls and Internet Security Repelling the Wily Hacker*, Addison-Wesley, 1994.
- [STO89] C. Stoll. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Doubleday, 1989.
- [SHI95] T. Shimomura. *The IP Spoofing Attack*, in Proceeding of the Third Workshop on Future Directions in Computer Misuse and Anomaly Detection, eds. Matt Bishop, Karl Levitt, and Biswanath Mukherjee. January 1995, appendix A-15.

Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions*

Mandy Chung

Nicholas Puketza
Biswanath Mukherjee

Ronald A. Olsson

Department of Computer Science
University of California, Davis, CA 95616
{chungm, puketza, olsson, mukherje}@cs.ucdavis.edu

Abstract

For testing Intrusion Detection Systems (IDS), it is essential that we be able to simulate intrusions in different forms (both sequential and parallelized) in order to comprehensively test and evaluate the detection capability of an IDS. This paper presents an algorithm for automatically transforming a sequential intrusive script into a set of parallel intrusive scripts (formed by a group of parallel threads) which simulate a concurrent intrusion. The main goal of parallelizing an intrusion is to distract an IDS's attention away from the intrusive activity. We identify constraints on the execution order among commands, and the way commands can be classified based on the effect of their execution. Synchronization and communication mechanisms are used to guarantee that the execution order among commands is preserved even under the parallelized scenario. We show that, experimentally, our work constitutes a major part of testing the ability of an IDS to detect intrusions and is especially useful for the users and developers of IDSs. We show that an intrusion is less likely to be detected if the suspicious activity is distributed over several sessions. Finally, we discuss some aspects of parallelizing intrusive scripts, including some practical difficulties that are open problems for future research.

Keywords: Intrusion Detection, Concurrency, Testing, Parallelization, Synchronization, Data Flow Analysis, Dependence Analysis.

1 Introduction

Intrusion detection provides a practical alternative approach to computer security besides designing a secure system [6, 12]. Intrusion Detection Systems (IDS) have been under investigation for many years [7, 14] and have started to move from laboratories to the real world. There is thus a need for sound methodologies and tools for testing IDSs. This paper presents our continuing ef-

fort on testing Intrusion Detection Systems [13].

We are researching methods for testing IDSs. In our testing experiments, we simulate intrusive activity, and then study the corresponding output from the IDS. We have developed a software platform that can be used to create scripts that simulate both normal and intrusive activities. We have also developed mechanisms in the platform to support concurrent intrusion simulations, including mechanisms for synchronization and communication (message passing) among different processes.

A major challenge of our work is to be able to simulate intrusions in various forms so that we can test an IDS's capability to detect intrusions comprehensively. A single intrusion can be executed in many different ways. For instance, an intruder may type in the intrusive commands one by one from a single terminal, or an intruder may code them up in a script. An advanced intruder may partition (or parallelize) the commands and issue them from different sources (e.g., different login sessions) to reduce the noticeability of the intrusion by an IDS. Similarly, multiple intruders may attempt to conceal an intrusion attempt by distributing the suspicious behavior amongst themselves.

Manually transforming a sequential intrusion into a concurrent one is very tedious and time-consuming. Besides, a single intrusion can typically occur in a number of different concurrent forms. For this reason, we envision, during the course of testing IDSs, the need for an automated approach to fragment a sequential intrusive script into parallel scripts that cooperate with one another.

This paper presents an algorithm for parallelizing a sequential intrusive script of Unix shell commands. Parallelizing an intrusive script has some similarities to parallelizing a program, which has been studied in depth [2, 3, 4, 5, 8, 9, 11]. Our work adapts some basic techniques used in program parallelization to fit in our context, including data flow analysis, dependence analysis,

*This work has been supported by the National Security Agency (NSA) INFOSEC University Research Program (URP).

and control dependence to data dependence conversion. Our algorithm is based on a dependency graph that represents the meaning of commands as well as their interrelations. A sequential intrusive script is first analyzed to determine various kinds of dependencies among commands which, in turn, enable us to determine their execution order. The sequential script is then transformed into a parallel script, in which synchronization and data communication mechanisms are employed to enforce the dependence relations of commands. A parallel script is formed from a group of parallel threads generated from our algorithm. By assembling parallel threads in different ways, various parallel forms of an intrusive script can be generated. This paper emphasizes the parallelization of intrusive scripts for testing IDSs; however, this approach can be employed for parallelizing arbitrary (non-intrusive) scripts, perhaps for different goals and different optimization.

The rest of this paper is organized as follows. Section 2 presents some example scenarios to illustrate the importance of an IDS to be able to detect concurrent intrusions, which raises the motivation and the need for our work. Section 3 presents some initial results from testing both sequential and parallel intrusive scripts on an actual IDS showing that intruders could escape detection by an IDS by distributing their intrusive activity over several concurrent sessions. Section 4 describes the model for our parallelization algorithm. Section 5 describes the steps involved in the automated parallelization (or transformation) of an intrusive script and also presents an algorithm for generating parallel threads. Section 6 discusses several aspects of parallelizing intrusive scripts, including some practical difficulties which we will deal with in the future. Section 7 concludes the paper.

2 Scenarios

This section presents two intrusion scenarios — password-guessing and password-cracking — demonstrating the fact that an intruder can possibly defeat an IDS's detection mechanism by issuing the intrusive commands from different sources concurrently (e.g., from different login sessions).

In the first scenario, an intruder attempts to guess the password of a user on a target machine. Obviously, he/she can do this by attempting repeated logins with different passwords until he/she successfully enters the system or until he/she gives up after several attempts. An IDS could probably detect this intrusion because a number of failed login attempts from a source machine is very noticeable. However, the guessing of passwords can be distributed among several intruders so that a group of passwords are tested simultaneously, perhaps from different machines. An intruder may also be able to manage to test several passwords concurrently through several open login sessions, or perhaps by an automated script. First,

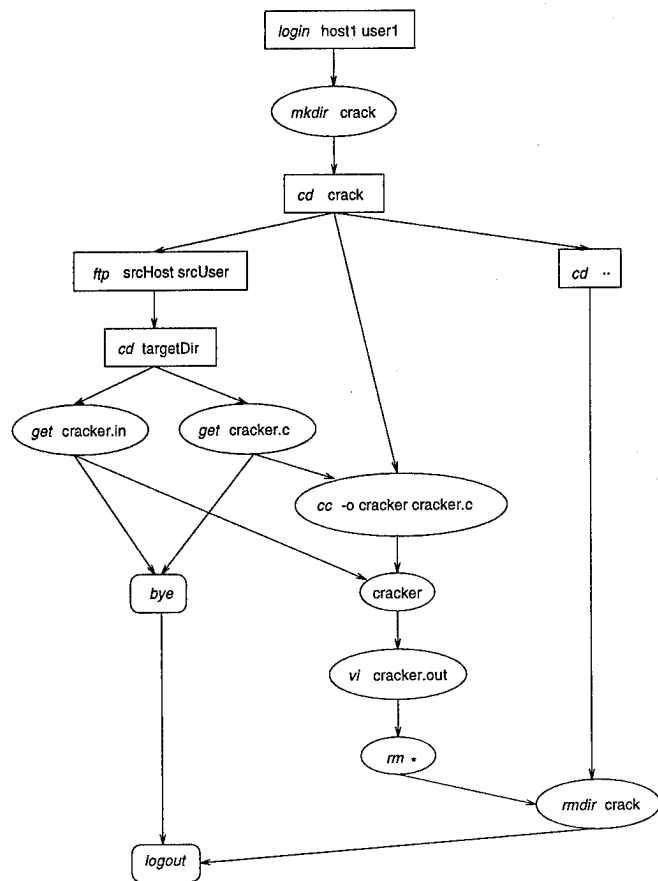


Figure 1: Dependence graph for a sequential password-cracking intrusion.

in this concurrent intrusion, the intruder can test all passwords in a shorter period of time. Second, this concurrent intrusion is less suspicious to an IDS than the sequential one because the logins are issued from different sources. Finally, if the target machine shares the password file with some other machines¹, the intruder can also test passwords on different machines simultaneously (rather than on one target destination). It is extremely difficult for an IDS to aggregate the activities issued from different sources to different destinations and to detect the coordinated intrusion.

In the second scenario, an intruder who manages to enter a target machine attempts to find any password that can be easily cracked. The intruder first logs into a target host *host1* as *user1* and then creates a temporary directory called *crack* under his/her home directory. He/she copies the password cracking program *cracker.c* and the dictionary file *cracker.in* from another machine *srcHost* and in a directory *targetDir* under *srcUser* home directory by *ftp*. Running the program *cracker* compiled from *cracker.c* will generate those cracked passwords into

¹This is usually the case when Network Information Service (NIS) is running, where the password file is shared among all NIS clients.

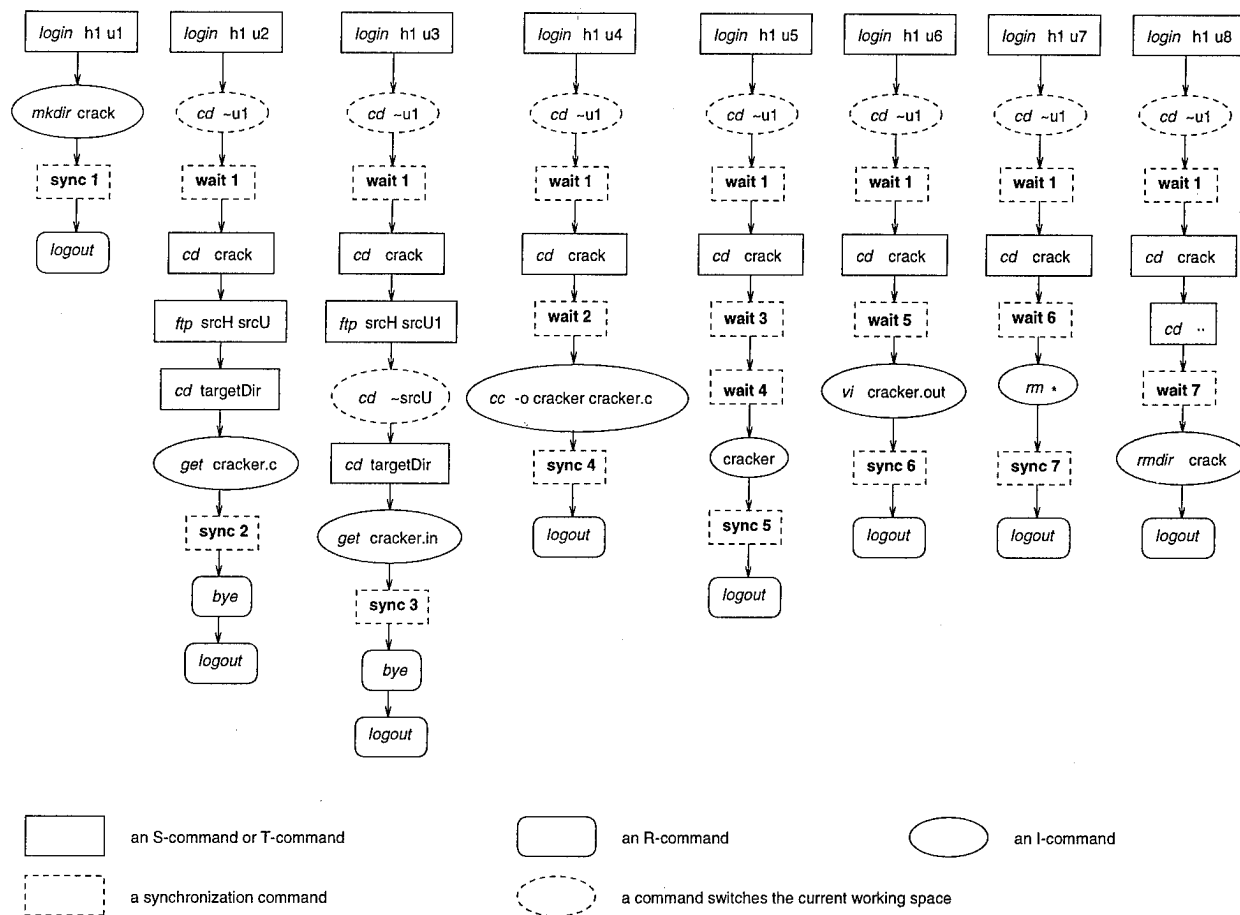


Figure 2: Parallelized password-cracking intrusion.

file cracker.out. After reviewing the output file, the intruder cleans up the working directory crack and leaves the session.

Although the actions in this intrusion, described above, seem to be necessarily performed serially in a single session, they can still be divided and performed in concurrent sessions by multiple coordinated intruders (or intrusion scripts). To illustrate, consider the commands executed in this intrusion and the dependence relations among them. Figure 1 depicts the dependence graph for this password-cracking intrusion. (In Figures 1 and 2, we distinguish between commands of four different types: S-command, R-command, T-command, and I-command, which will be described in Section 4.) Figure 2 shows one possible parallel version of this intrusion. It consists of eight individual intrusive sessions running simultaneously. Each of these intrusive sessions carries minimal activity as shown. That is, it is not possible to separate into two or more threads the activity performed in any one of these parallel threads. Concerning the benefits of parallelizing this password-cracking intrusion, the parallel version does not gain any considerable speedup. On the contrary, the elapsed time of the intrusion may be increased due to the overhead of synchronization be-

tween the various parallel threads. However, the speedup is not the main goal of the intrusion parallelization operation. The main goal is to disguise the intrusive activity performed by an intruder or a group of intruders. The major benefit obtained from parallelizing the password-cracking intrusion is to distribute the intrusive activity among concurrent sessions so as to minimize the chance of detection of the activity.

The two examples described above both involve the interaction between the intruder and the shell. Another form of intrusion could be due to a program that makes system calls. Activities issued from either of them may be collected by an audit trail on which the analysis of many IDSs rely. In this paper, we focus on the former form of intrusion involving shell-level commands.

3 Experimental Results

To escape detection by an IDS, intruders might try to distribute their intrusive activity over several concurrent sessions. The premise behind this strategy is that the IDS will assign a higher warning value to one *very* intrusive session than it will to several less intrusive sessions. We conducted some experiments to test this premise.

The IDS that we tested is the Network Security Monitor (NSM) [7]. The NSM monitors all of the packets that travel on the local area network (LAN) to which the NSM host computer is connected. The NSM can associate each such packet with the corresponding computer-to-computer connection. It assigns a warning value between 1 and 10 (higher is more suspicious) to each connection based on the contents of the packets, and on the likelihood of the connection occurring, given a record of recent connections. We ran the NSM on a Sun Sparc Station 2 workstation connected to the Computer Science LAN segment at UC Davis (UCD).

3.1 Test Procedure

We first selected four intrusive activities:

1. transmitting the */etc/passwd* file from one computer to another;
2. password-cracking by comparing the entries in the */etc/passwd* file to a list of encrypted password guesses;
3. password-guessing using a dictionary file; and
4. exploiting the *loadmodule* flaw to achieve super-user status.

For each of these activities, we created a sequential script to simulate the activity. Then, we manually created a concurrent script set which collectively included all of the commands from the sequential script. We activated the NSM and ran the scripts. We then compared the warning values for the sequential script with the warning values for the concurrent script set. The results are displayed in Figure 3. The NSM assigns a warning value to each network connection. Several of the scripts and script sets initiate more than one network connection, but for clarity the figure shows only the maximum warning value for all network connections associated with each script and script set.

INTRUSION DESCRIPTION	SCRIPT TYPE s = sequential c = concurrent	MAX WARNING VALUE
transmitting passwd file	s	7.472
	c	7.472
password-cracking	s	3.160
	c	3.160
password-guessing	s	8.722
	c	7.785
exploiting loadmodule flaw	s	7.472
	c	4.972

Figure 3: NSM experimental results.

3.2 Analysis of Results

For the first intrusion simulation in Figure 3, the warning value for the concurrent script set is the same as the warning value for the sequential script, and the warning values are high. A possible explanation for this is that the sequential script contains a very suspicious command or set of commands which cannot be divided when the concurrent script set is created. As a result, at least one of the threads in the concurrent script set is by itself just as suspicious as the original sequential script. The warning values for the second intrusion simulation are again equal, but in this case the values are low. A likely explanation is that the NSM was not configured to be sensitive to that particular intrusion. So, neither the sequential script nor the concurrent script set produced activity that appeared suspicious to the NSM.

For each of the last two intrusion simulations in our experiments, the warning value for the concurrent script set is less than the warning value for the sequential script. In both cases, it was possible to divide up a set of suspicious commands in the sequential script among two or more threads in the concurrent script set.

Taken together, our experiments indicate that it is possible for intruders to reduce the chance of detection by an IDS by distributing their suspicious activities, although this strategy is not always successful. In future work we plan to investigate the effects of this strategy on different IDSs. For example, the NSM monitors each network connection independently. An IDS that, instead, keeps track of all the activity associated with each user may not be affected as much by this intruder strategy.

4 Model

Our work focuses on the automated parallelization (or transformation) of an *intrusive script* that is used to simulate an intruder's activity. An intrusive script is written in a simple programming language which allows us to specify shell-level commands, such as shell language [15] and *Expect* [10]. In addition, the language typically includes variables, procedure, and control-flow statements, such as if-then-else and loop.

This section presents a model for an intrusive script transformation which focuses on issuable shell-level commands in an intrusive script. The model is divided into two parts: shell-level commands and statements in an intrusive script.

An intrusion is considered to be a sequence of shell-level commands issued by an intruder. An intruder can issue commands one by one from a single terminal, or issue commands from more than one terminal. For example, an intruder can create two login sessions from two different windows on his/her workstation to a target host at the same time. On the target host, the identity of the user associated with these sessions can be different

(if the intruder manages to get two different accounts on that system). In this paper, we refer to the commands that are issued from a terminal as an *intrusive session*. More specifically, a user can consecutively issue several commands that create a new user session, such as *rlogin*, *telnet*, and *ftp*, from a terminal in which a hierarchical structure of open user sessions is built. An intrusive session refers to the root session of this structure. A sequential intrusion involves only one intrusive session while a concurrent intrusion typically involves multiple intrusive sessions.

A shell-level command, like a procedure, can take parameters (e.g., from command line) and return a result (e.g., to standard output). In addition, a command may change two kinds of states in a computer system: the file system state and the intrusive session state. The file system state includes the existence and the content of files in the file system. The intrusive session (IS) state consists of all predefined and user-defined environment variables, including the real user ID (uid), the effective user ID (euid), the group ID (gid), the current working directory (cwd), and the hostname. The current IS state refers to the state of the active user session in an intrusive session. We characterize a command *C* by the following:

- Input parameters and output result.
- A set of file system objects from which *C* reads.
- A set of file system objects to which *C* writes.
- A set of IS state attributes² on which *C* depends.
- A set of IS state attributes which *C* changes.

Based on the above definition of a command, the dependence constraints on the intrusive script transformation are defined as follows:

Data dependence. Two commands are data dependent if the input parameter of one command is determined by the output of another, or some file system objects written by one command are referenced (read or written) by another and these file system objects are not written by other commands between the execution of these two commands.

Attribute dependence. Two commands are attribute dependent if some IS state attributes changed by one command are referenced (read or changed) by another and these attributes are not changed by other commands between the execution of these two commands.

An intrusive script typically consists of various constructs provided by the language. Control dependence is another constraint to the parallelization problem due to the presence of control-flow statements in the script (see Section 5.2.1 for details).

²In order to distinguish between an environment variable and a script variable, we refer to an environment variable as an attribute of the IS state.

We also classify commands into four different types according to their effect on the IS state:

1. I-command (*state Invariant*) — A command that does not affect the IS state. For example, *ls*, *cp* and *cat* are I-commands. These commands only affect the file system state, e.g., change the content of a file or create a new file.
2. S-command (*Session creation*) — A command that creates a new user session and changes the IS state, but the IS state before executing this command can be restored by an R-command. Examples of S-commands are *rlogin* and *ftp*, which change the IS state attributes, uid, euid, gid, cwd, and host. *su* is slightly different from the above since it only changes uid, euid, and gid.
3. R-command (*state Restoration*) — A command that closes the current user session and restores a previous IS state. Specifically, an R-command reverts the IS state to a state in which the corresponding S-command began without requiring the knowledge of those executed commands and their parameters, or the value of the previous IS state. For example, *logout* is an R-command corresponding to *rlogin* whereas *bye* is an R-command corresponding to *ftp*.
4. T-command (*state Transition*) — A command that changes the IS state, but no R-command corresponds to it. For example, *cd* is a T-command that changes the current working directory (cwd) and *setenv* is another T-command that defines an environment variable. Although the IS state before executing a T-command can be restored via a series of T-commands, *cd* and *setenv* are neither an S-command nor an R-command since they require the knowledge of the value of a previous IS state (for restoration) and since they do not open or close a user session.

As we described earlier, a script can contain variables, control-flow statements, and constructs for issuing shell-level commands. Figure 4 shows a simple *Expect* script that controls an *rlogin* session.

For simplicity, when parsing the script statically, we refer to a statement that issues a shell-level command as a “*command*” (e.g., lines 5, 7, 11, 13, and 14 in Figure 4), while we refer to other constructs provided by the language as “*statements*” (e.g., lines 2, 3, and 10 above). More precisely, the issuable commands (line number of its referred statement) in Figure 4 are: *rlogin* (lines 5 and 7)³, *whoami* (line 11)⁴, *ls -l* (line 13), and *logout* (line 14). Lines 10 and 11 together form a conditional statement containing an issuable command *whoami*.

³Inputting password is considered to be part of the *rlogin*.

⁴Line 11 will be invoked only if *rlogin* to the host occurs as root.

```

1 # get target host and user from arguments
2 set host [lindex $argv 1]
3 set user [lindex $argv 2]
4 # spawn an rlogin process
5 spawn rlogin $host -l $user
6 # expect the password prompt,
  # then send the password.
7 expect {"Password:" send "actualpassword\r"}
8 # expect the shell prompt,
  # then send shell-level commands
9 # The shell prompt is specified
  # in a regular expression.
10 if {$user == root} {
11     expect {-re ".*%|. *>|. *#" send "whoami\r"}
12 }
13 expect {-re ".*%|. *>|. *#" send "ls -l\r"}
14 expect {-re ".*%|. *>|. *#" send "logout\r"}

```

Figure 4: A simple *Expect* script.

5 Automated Parallelization of an Intrusive Script

Parallelization of a sequential intrusive script consists of the following steps :

1. Parse an intrusive script and build a flow graph.
2. Convert the control dependence to data dependence.
3. Perform dependence analysis and build a data dependence graph.
4. Create parallel threads for the intrusive script and insert synchronization and data communication commands to facilitate coordination between parallel threads.
5. Perform optimization and transformation, if any.
6. Generate a parallel intrusive script.

A flow graph representing an intrusive script is different from a flow graph representing a program [1]. We define a basic block, in our context, as a sequence of consecutive “statements” and one issuable “command”. Therefore, it is possible to have a basic block containing a conditional statement or loop which contains no command. The intrusive script transformation procedure is only interested in the issuable shell-level command contained in a basic block. We assume that two basic blocks are dependent only if the issuable commands in the basic blocks are dependent (see Section 4). Information obtained from the evaluation of other statements in a basic block is only used within the basic block and is independent of other basic blocks. That is, script variables used or modified in a basic block are not referenced elsewhere.

In parallelizing an intrusive script, we must obey the constraints of the underlying dependence structure of the script. In our algorithm, a data dependence graph [8] is

used to represent the data dependence, attribute dependence, and control dependence. Both data and attribute dependence of commands can be represented in a data dependence graph because the IS state attributes are another form of data in an intrusive session. We can also treat control and data dependence uniformly by applying a technique in parallelizing compilers introduced by Allen and Kennedy [2] to convert control dependence into data dependence. We also adapt the dependence analysis [3, 5, 8, 9] used in program parallelization to determine the dependence relations of all issuable commands in a script.

In the following, when we refer to a command in a dependence graph, we actually mean the basic block containing this command. A node in a dependence graph represents a basic block which contains one issuable command and other statements.

Section 5.1 presents an algorithm to generate parallel threads in parallelizing an intrusive script, which does not have branch, loop, and procedure. Section 5.2 extends the algorithm to handle conditional statements and loops. It also discusses another dependence due to the script variables used or modified in a basic block and referenced by another. Section 5.3 discusses the possible optimization and transformation performed on the parallelized intrusive threads.

5.1 Parallelization Algorithm

This algorithm consists of two phases: parallel threads generation phase and threads synchronization phase. Appendix A gives the pseudo-code for this algorithm.

5.1.1 Parallel Threads Generation Phase

We represent an intrusive script as a series of commands, $I = \{C_1, C_2, \dots, C_n\}$ since there is no branch, loop, and procedure in the script. The IS state transition during the execution of I is $\{s_0, s_1, \dots, s_n\}$ where s_0 is the initial IS state determined by the input of I . This phase creates a parallel thread for each I-command in I . In order to guarantee that an I-command executed in a parallel thread has the same effect as in the sequential script, the IS state s_{i-1} must be reached before the execution of C_i begins. Specifically, s_{i-1} is reached by executing all S-commands and T-commands in $\{C_1, C_2, \dots, C_{i-1}\}$ serially.

The details of the algorithm are as follows. The algorithm processes each command and uses a stack to store those S-commands and I-commands whose executions reflect the current IS state. When processing an I-command, the algorithm creates a new thread for performing this command. We use a flow graph to represent a sequence of commands executed in a parallel thread. First, a flow graph is formed by creating a node for each element (each command) in the stack; the bottom one in the stack is the first command executed in the

thread while the top one is the last command. The algorithm then appends a node representing the I-command to the graph. When processing an S-command or a T-command, this command is pushed onto the stack and no thread is created. If it is an S-command, all new threads containing this S-command are recorded so that their open sessions can be closed appropriately. When processing an R-command C_r , all recorded threads for the S-command C_s that corresponds to C_r are appended with C_r to close their current sessions. Precisely, C_s is the top S-command in the stack. All subsequent commands following C_s (including C_s) in the stack are popped (since the open session created by C_s is closed by C_r).

After processing all commands in the script, the number of parallel threads generated is the number of I-commands in the script. For example, Figure 1 is the dependence graph for the sequential password-cracking intrusion scenario discussed in Section 2, and it turns out that this example has eight I-commands. Figure 2 shows the eight parallel threads generated by this algorithm and each thread performs only one of these eight I-commands.

5.1.2 Threads Synchronization Phase

After all parallel threads are generated, the dependence relations among commands are enforced in this phase to guarantee the execution order of the commands by inserting synchronization and data communication mechanisms [11].

A dependence graph G consists of nodes and directed edges. A node represents a basic block containing a single command. A directed edge (u, v) represents a dependence relation between basic blocks u and v , i.e., the execution of v can begin only after the execution of u terminates. By performing breadth-first search on G , all commands (nodes) are visited. While visiting a node containing an I-command C_i , we insert a synchronization command to each successor of C_i in G , C_j , to ensure that its execution begins only after C_i terminates. If the input of C_j depends on the output of C_i , the output of C_i is sent to the thread containing C_j using data communication commands.

As in the parallel threads generation phase, all S-commands, T-commands, and R-commands may be duplicated in other threads whereas an I-command is executed in only one parallel thread. If C_i is either an S-command, a T-command, or an R-command, the execution order of C_i and its successors is guaranteed to occur in sequential order.

5.2 Language Constructs

5.2.1 Conditional Statements

Conditional statements are very useful in simulating an intrusion. As a simple example of control dependence in an intrusion, consider an intruder who checks the permission mode of a file named fileA and decides whether

to read fileA or to modify fileA. The first command performed by the intruder is `ls -l fileA` to obtain the file access information. According to the file access permissions of fileA, if it is world-writeable, he/she then modifies the file by `vi`; otherwise, if it is world-readable, he/she reads the file by `cat`.

For simplicity, in the following example, we focus on the issuable commands and ignore all other statements within the if-statement.

```

if B then
    command1
    command2
else
    command3
endif
command4

```

Command1, command2, and command3 are control dependent on the boolean expression B since B determines which command is executed. (B may be obtained from the output of a previous command.) The conversion from control dependence to data dependence proceeds by first replacing the if-statement at the source of the dependence with an assignment statement to a boolean variable. The converted if-statement is:

```

b = B
command1 when b
command2 when b
command3 when not b
command4

```

All control dependent commands are tagged with a "when b " or "when not b " clause depending on to which arm of the original if-statement they belong. The operator **when** indicates that the expression on its left is executed only if the boolean expression on its right is true. A boolean variable b is used instead of B because statements in either arm of the if-statement could have side effects, i.e., these side effects could change B. After the conversion, the control dependences of command1, command2, and command3 become flow dependences generated by variable b and each command is contained in a single basic block in the flow graph.

After the if-conversion, the parallel threads can be generated as described in Section 5.1 but a slight modification must be made to handle the tagged R-commands as follows. When processing a tagged R-command with operand b , C_r **when** b , all recorded threads containing its corresponding S-command C_s are appended with this tagged R-command. Consider those S-commands and T-commands executed between C_s and C_r , i.e., $\{C_s = C_{i_0}, C_{i_1}, \dots, C_{i_k}\}$ in the stack. First, for $0 \leq j \leq k$, all tagged C_{i_j} whose operand is satisfied with the current b value are popped from the stack because, if b is true, the

current user session is closed. Then, all remaining C_{i_j} in the stack are modified to tag with a "when not b " clause if it is not tagged, or to replace the "when c " clause with a "when c and not b " clause. Therefore, all remaining C_{i_j} in the stack are guaranteed to execute under correct IS state.

The tagged commands now depend on the basic block that evaluates b . All threads that contain the tagged command may need to receive the b value from another thread, which we refer to as variable dependence (to be discussed in Section 5.2.3).

5.2.2 Loops

Recall from Section 2 that, in the sequential password-guessing intrusion example, an intruder repeatedly attempts to log into a target machine and guess a password until he/she successfully enters the target machine or finishes guessing all passwords in his/her stock. This intrusion obviously contains a loop for testing passwords.⁵ In this example, an iteration of the loop — logging in and guessing a password — is independent of other guesses and can be executed concurrently. Although loops in an intrusive script may not be as commonly used as loops in a program, this construct is considerably useful in simulating certain types of intrusions.

We follow the terminology used in parallelizing compilers proposed by Banerjee [3] to classify three parallel loop forms.

- DOALL is a loop that allows total parallel execution, i.e., all iterations of the loop body are allowed to run simultaneously.
- DOACROSS is a loop that allows partial overlap of successive iterations during execution.
- DOSEQ is a sequential loop without parallelism.

After performing the loop dependence analysis, we can identify the type of loops contained in the script. Two kinds of DOALL loops are parallelizable. First, a DOALL loop that contains only I-commands is parallelizable. Second, a DOALL loop is parallelizable if the loop's body can be divided into three blocks such that the first and the last blocks contain only I-commands while the middle block starts with an S-command and ends with an R-command corresponding to this S-command. In other words, a parallelizable DOALL loop terminates with a previous IS state right before its execution begins. Because each iteration of the loop is independent on others, a new thread is generated for executing an iteration of the parallelizable loop. A special type of node is used to represent this parallelizable DOALL loop in the dependence graph so that the parallel threads generation phase can recognize this loop and parallelize it accordingly.

⁵When the connection is closed (after the configured number of incorrect login attempts), the intruder may need to issue a login command again.

DOACROSS loops containing only I-commands (no IS state change) can also be parallelized in a similar manner but synchronization and data communication mechanisms must be appropriately inserted both at the end of one iteration and before another iteration begins, just as in processing dependent commands.

If a thread generated from the loop parallelization procedure contains more than one I-command, this thread can further be parallelized as if it is a sequential intrusive script by recursively applying the algorithm on it. Parallel threads may be required to transmit data due to the loop parallelization because of DOACROSS loops.

Other kinds of loops are considered as non-parallelizable. Among these non-parallelizable loops, a loop containing only I-commands is treated as a single basic block so that a single thread can be created to perform this loop's activity. However, a non-parallelizable loop that contains commands other than I-commands will inhibit the parallelization. That is, all commands following this loop together with the loop is treated as a single basic block in the dependence graph, and hence, it is the last thread created in the parallelization. In particular, a script that has this kind of loop but no I-command before it is non-parallelizable. In fact, under some situation, further parallelization may be allowed even if the script contains such non-parallelizable loops. For example, if the execution of a loop restores the original IS state after it terminates, it can be treated as a single basic block. Duplication of the loop in each parallel thread generated for commands that follow it may also be feasible in some cases. However, the analysis involved to guarantee that the execution of such a non-parallelizable loop in more than one thread resulting in a correct and safe state is very complicated and difficult.

5.2.3 Variable Dependence

So far, we have considered that information obtained in a basic block is independent of other basic blocks. Two basic blocks are dependent only if the issuable commands in the basic blocks are dependent. Therefore, script variables used or modified in one basic block are not referenced in the others. Under certain circumstances, such as the operand introduced by the if-conversion, the algorithm may need to handle the variable dependence across basic blocks.

5.3 Optimization and Transformation

After generating parallel threads from the dependence graph, several possible optimizations can be performed on each thread. One possible optimization is to log in as a different user in each thread. Consider the password-cracking example presented in Section 2. Figure 2 shows eight parallel threads generated from the dependence graph shown in Figure 1 by the algorithm. As shown in Figure 2, each thread can *login* as a different user un-

less user1 is *root* in the sequential script (Figure 1). If different users are used in parallel threads, we must ensure that all referenced files are accessible by all of these users. This can be achieved either by setting the permission mode of these shared files explicitly by the users, or by assigning one thread to change the permission modes of these shared files.

Another possible optimization is to remove redundant commands executed in a thread. For example, the last parallel thread in Figure 2 removes the temporary directory *crack*; however, two redundant *cd* commands are executed before *rmdir*. Obviously, they can be removed from this thread without altering its intrusive behavior.

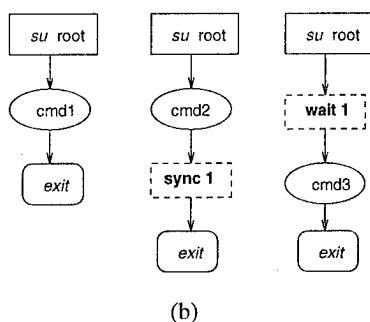
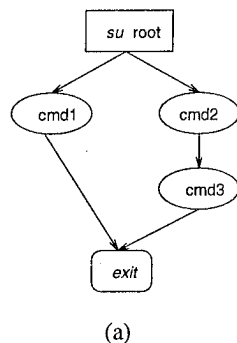


Figure 5: Transformation example.

As the parallel threads generated contain minimal activity, a login session in the sequential intrusion may be broken down into multiple login sessions in the concurrent intrusion. For example, Figure 5(a) shows a subtree of the dependence graph representing one part of the intrusion and Figure 5(b) shows the three parallel threads generated to perform the activity in Figure 5(a). Although these parallel threads perform individual root login session independently, three root logins may be more suspicious to an IDS than the one root login in the sequential intrusion. In this case, parallelization of this subtree may not be beneficial. One possible transformation in this example is to combine the threads into one thread to avoid creating suspicious login sessions. Therefore, Figure 5(a) is used for that part of the intrusion after the transformation.

The parallel intrusion generated by our current approach uses the same working space as the sequential in-

trusion. For instance, in Figure 1, all files created or added by this sequential cracker intrusion are placed under the directory *~user1/crack* in machine *host1*, and they will be removed at the end of the intrusion. Although different parallel threads of the concurrent cracker intrusion can log into different users and have their own working space, they access and use the same working space as in the sequential intrusion, e.g., the directory *~user1/crack*. With further analysis on the semantics of the script, it might be desirable to transform the threads to use their own working space if possible.

6 Discussion and Future Work

This section discusses several aspects of parallelizing an intrusive script: various forms of parallel scripts generated by our algorithm, some practical difficulties, and generalization of our algorithm.

We have presented an algorithm for parallelizing a sequential intrusive script into one possible parallel intrusive script in Section 5. The intrusive activity is basically performed by several parallel threads concurrently. Each parallel thread carries minimal intrusive activity. In fact, other possible parallel intrusive scripts can easily be generated by assembling parallel threads together in different ways to form different combined threads. Thus, an intrusion can be systematically mutated from a sequential form into various different parallel forms, which can be used in testing an IDS.

Parallelizing an intrusive script is difficult in practice because of the rich set of shell-level commands, and various constructs supported by the script language. However, we will deal with these difficulties and search for their possible solutions in the future. Five practical difficulties are discussed below.

Domain and range analysis of shell-level commands. Our algorithm relies on the assumption that we have the knowledge about the domain and range of every Unix command. The domain of a command is the set of file system objects it reads from and the set of IS state attributes it uses, while the range is the set of file system objects it writes to and the set of IS state attributes it changes. However, they are difficult to obtain systematically. First, the domain and range of a command may differ with different arguments. Sometimes, the meaning of a command changes substantially with different option arguments, and so do the domain and range. For instance, the command "*cp a b*" copies the file *a* to a file *b*, whereas the command "*cp -r a b*" copies all files under the directory tree rooted by *a* to a directory *b* if *a* is a directory. In this example, the domain and range of these two *cp* commands may be different due to the "*-r*" option. In addition, the user is able to redirect I/O to and from files as well as redirect the output of one command as input to another command using pipes. For example, the command "*ls -l > file*" changes the file system state

while the command "*ls -l*" alone does not. In Unix, command aliases, hard links, and symbolic links are allowed to be created. The command *cp* at one time may refer to the program */bin/cp*, but at another time, it may refer to */bin/ls* if the user has made an alias named *cp* of this *ls* command. Similarly, references on a hard link or a symbolic link made by a command may actually refer to another file. Apart from standard Unix commands, a user can also execute a user-defined command which can be a program making system calls. To determine the domain and range of such a program requires detailed analysis of the program source.

Complexity of text editor commands. Our algorithm cannot completely handle text editor commands because of their complexity. For example, *vi* can edit different files before exiting the editor session. It can also start a new user session via shell escape. Activities performed within an editor session are very hard to analyze from a script. We currently handle *vi* and *emacs* as I-commands that only access and modify one file specified as a parameter and do not have other side effect.

Side effect of additional open user sessions. S-commands and T-commands may be duplicated in parallel threads. The execution of the duplicated S-commands on parallel threads opens additional user sessions which may affect the current state of the computer system, such as the list of users currently on the system, the number of active processes, and the last login time of an attacked account. In other words, execution of S-commands may affect the output of some I-command in a script. Our algorithm does not currently handle an intrusive script containing an I-command which depends on the system state. For example, one of the actions performed by an intruder is to discover the last user who logs into machine A. When an intruder has logged into A, the user whom the intruder wants to find becomes the second last user who logs into A. The command "*last -2*" can be used to get this information in a sequential intrusive script, but it is not necessarily correct when used in a parallel script.

Interprocedural Analysis. Our algorithm handles control-flow constructs used in a script, such as if-then-else and loops. Procedures are another construct that complicates our dependence analysis. However, by analyzing the effect of a procedure call, including which parameters changed on return, what global variables are used and modified, and other side effects, we can determine whether the procedure presents a constraint on an intrusive script's parallelization. In some situations, a procedure can be expanded in-line by substituting the formal parameters with actual parameters, and renaming local variables. However, in-line expansion is not applicable to recursive procedures.

Suspension and resumption of open user sessions. Currently, our algorithm does not handle commands that

suspend or resume a user session. We believe that these commands are rarely used in intrusions. Our algorithm can easily be extended to handle them. The command that suspends a user session can be treated similar to an R-command. Instead of just restoring a previous IS state, those commands whose executions reflect the suspended IS state are kept so that this suspended IS state can be restored when this suspended user session is resumed.

Our algorithm described in this paper focuses on parallelizing a sequential intrusive script of Unix shell commands. However, it can be generalized to apply to other system platforms, e.g., VMS. The major difference in the intrusive script transformation on different system platforms is the definition of the intrusive session state on which the dependence analysis depends, but same kinds of analysis can be used. In the future, we will study the dependence relations of commands on various platforms and thus our work can be applied to testing other IDSs that run on other system platforms.

7 Conclusion

In this paper, we have presented an automated mechanism for parallelization of intrusive scripts for testing an IDS. Being able to simulate an intrusion in different forms is very important for testing the ability of an IDS to detect intrusions. Parallelizing an intrusive script has some similarities with parallelizing a program; however, they differ in some aspects mainly due to the shell-level commands involved in a script and their additional constraints on parallelization. By modeling shell-level commands and intrusive scripts, we can adapt the methodologies used in program parallelization for parallelizing intrusive scripts.

The transformation of an intrusive script allows us to generate other possible parallel forms so that an IDS can be thoroughly tested. We conducted some experiments on testing an IDS with both sequential and parallel intrusive scripts. The initial results showed that the detection mechanism of an IDS could be defeated when an intruder distributes the intrusive activity over concurrent sessions. We believe that our work is especially useful to the developers of IDSs in testing their products. Besides, a system administrator can also evaluate and compare the effectiveness of the detection mechanism of an IDS with the help of the intrusive scripts transformation. We expect that our work constitutes a major part in testing IDSs.

In the future, we will deal with the practical problems in parallelizing intrusive scripts that we have discussed. We will also conduct some experiments on testing operational IDSs with both sequential and parallel intrusive scripts to obtain further results. Finally, we would like to mention that our work is used for testing IDSs rather than for launching new intrusions.

References

- [1] A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers: Principles, Techniques, and Tools.*, Addison-Wesley, 1986.
- [2] J.R. Allen, et al., "Conversion of Control Dependence to Data Dependence", *Proceedings of 10th Annual ACM Symposium on Principles of Programming Languages*, Austin, Texas, January 1983.
- [3] U. Banerjee, *Dependence Analysis for Supercomputing*, Kluwer Academic Publishers, Boston, Mass., 1988.
- [4] U. Banerjee, R. Eigenmann, A. Nicolau, and D.A. Padua, "Automatic Program Parallelization", *Proceedings of the IEEE*, vol. 81, no. 2, pp. 211-43, February 1993.
- [5] M. Burkner and R. Cytron, "Interprocedural Dependence Analysis and Parallelization", *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction*, pp. 17-22, June 1984.
- [6] D.E. Denning, "An Intrusion Detection Model", *IEEE Transactions on Software Engineering*, vol. SE-13, pp. 222-232, February 1987.
- [7] 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, May 1990.
- [8] D. J. Kuck, et al., "Dependence Graphs and Compiler Optimization", *SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pp. 207-218, January 1981.
- [9] Z. Li, P. Yew, and C. Zhu, "An Efficient Data Dependence Analysis for Parallelizing Compilers", *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 1, pp. 26-34, January 1990.
- [10] D. Libes, "Expect: Curing Those Uncontrollable Fits of Interaction", *Proceedings of the Summer 1990 USENIX Conference*, June 1990.
- [11] S. Midkiff and D. Padua, "Compiler algorithm for Synchronization", *IEEE Transactions on Computers*, vol. C-36, no. 12, pp. 1485-1495, 1987.
- [12] B. Mukherjee, L.T. Heberlein, and K.N. Levitt, "Network Intrusion Detection", *IEEE Network*, vol. 8, no. 3, pp. 26-41, 1994.
- [13] N. Puketza, B. Mukherjee, R.A. Olsson, and K. Zhang, "Testing Intrusion Detection Systems: Design Methodologies and Results from an Early Prototype", *Proc. 17th National Computer Security Conference*, vol. 1, pp. 1-10, October 1994.
- [14] S.E. Smaha, "Haystack: An Intrusion Detection System", *Proc. IEEE 4th Aerospace Computer Security Applications Conference*, December 1988.
- [15] P. Wang, *An Introduction to Berkeley Unix*, Wadsworth Publishing Company, Belmont, California.

A Appendix: Algorithm for Parallelizing Intrusive Scripts

Parallel Threads Generation Phase:

```

S = empty stack /* S-commands stack */
M =  $\emptyset$  /* resulting parallel threads */
For each command  $C_i \in$  an intrusion I do
  if  $C_i$  is an S-command with uid = U then
    o push(S,  $C_i$ ); push(S, cd ~U)
  if  $C_i$  is a T-command then
    o push(S,  $C_i$ )
  if  $C_i$  is an I-command then
    o  $g = \text{gen.graph}(S)$ ; append  $C_i$  to  $g$ 
    o  $M = M \cup \{g\}$ 
  if  $C_i$  is an R-command then
    o find the S-command  $C_s$  in S that corresponds to  $C_i$ 
      (i.e., the top S-command in  $S = C_s$ )
    o for each graph  $g$  containing  $C_s$  do
       $\diamond$  append  $C_i$  to  $g$ 
    o pop(S) until  $C_s$  is pop

```

gen.graph(S) generates a new thread containing all commands in the stack S.

Threads Synchronization Phase:

Perform a BFS on the dependence graph G and perform the following when visiting a node of command C_i :

```

if  $C_i$  is an I-command then
  o find the graph  $g$  in M containing  $C_i$ 
  o for each successor of  $C_i$  in G, say  $C_j$ 
     $\diamond$  for each graph  $g'$  in M containing  $C_j$ 
      o if input of  $C_j$  depends on output of  $C_i$  then
         $\diamond$  insert command in  $g$  after  $C_i$ 
          (to send data to  $C_j$  in  $g'$ )
         $\diamond$  insert command in  $g'$  before  $C_j$ 
          (to receive data from  $C_i$  in  $g$ )
      o else
         $\diamond$  insert sync command in  $g$  after  $C_i$ 
         $\diamond$  insert sync command in  $g'$  before  $C_j$ 

```


MAINTAINING PRIVACY IN ELECTRONIC TRANSACTIONS

Benjamin Cox <thoth+@cmu.edu>

Information Networking Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213-3890

Abstract

Electronic commerce presents a number of seemingly contradictory requirements. On the one hand, we must be able to account for funds and comply with laws requiring disclosure of certain sorts of transaction information (*e.g.*, taxable transactions, transactions of more than \$10,000). On the other hand, it is often socially desirable to limit exposure of transaction information to protect the privacy of the participants.

In this paper, I address the following issues:

- I develop a new analysis technique for measuring the exposure of transaction information.
- I analyze various privacy and disclosure configurations to determine which are technically feasible and which are logically impossible.
- I apply this analysis to the proposed NetBill billing server protocol.
- I consider the use of intermediary agents to protect anonymity and the implications of various arrangements of intermediaries.

New contributions include a new analysis technique and its associated notation, a system for generating ad hoc pseudonyms to protect privacy, the application of message forwarding techniques to protecting privacy in electronic commerce, and the application of these methods to the NetBill system.

1. Introduction

The time is ripe for commerce over the Internet. With the advent of the World Wide Web, merchants of many kinds are seeing the advantages of making their wares available on the Internet. Along with the inrush of potential merchants comes an inrush of electronic commerce technologies; the NetBill system being designed at Carnegie Mellon University's Information Networking Institute is one of many systems in development. Others include a system of anonymous credit cards being researched at AT&T Bell Labs, the ECash system of digital currency being developed by DigiCash, the CyberCash system from CyberCash, Inc., the First Virtual Internet Payment System from First Virtual Holdings Incorporated, and the NetCheque system being developed at ISI. A major goal of NetBill is to reduce the transaction processing overhead to accommodate purchase prices on the order of ten cents per transaction. A main design feature of NetBill is that it uses a central server as an exchange point between merchants and consumers, rather than requiring merchants to have prearranged relationships with their customers.

NetBill's central-server approach has advantages and disadvantages. Some advantages are simplified authentication authority, single-statement billing, and simplified access to account information. Disadvantages include network and processor bottlenecks, and privacy concerns.

With a central billing server handling all details of transactions and providing authentication services to all parties, it is very simple to compile dossiers on consumers and merchants unless precautions are taken. When large compilations of personal information are readily available, the potential for abuse is great. Abuse could range from an explosion of direct marketing campaigns to use of the information to target groups of people as potential victims for criminal activity.

In this paper, I discuss mechanisms for hiding various pieces of the transaction information to maintain the privacy of users, both consumers and merchants. New contributions include:

- a new analysis technique and its associated notation
- a system for generating ad hoc pseudonyms to protect privacy
- the application of message forwarding techniques to protecting privacy in electronic commerce, and
- the application of these methods to the NetBill system.

2. The Transaction Information Matrix

It is useful to develop a model showing what portion of transaction information is available to which involved parties. This section presents a matrix notation indicating information disclosure.

2.1. Parties Involved and Available Information

For various reasons, we may wish to hide information from (or disclose information to) any of the following parties (some of whom are directly involved in the transaction, and some of whom are not): the merchant, consumer, the billing server, government authorities (such as tax authorities), any applicable auxiliary parties, and observers.

In a fully disclosed electronic transaction, information is available about the merchant's and the consumer's identities, account numbers and network addresses; the items purchased; and the transaction's price and tax status.

2.2. The Matrix

Table 1 shows the basic matrix indicating information disclosure. A symbol in a matrix block indicates disclosure of information to a specific party.

Table 1: Empty transaction information matrix.

	Consumer's			Merchant's			Items	Amount	Tax Status
	identity	account	address	identity	account	address			
Consumer									
Merchant									
Billing Server									
Authorities									
Auxiliaries									
Observer									

The following symbols are defined:

- X: The information is fully disclosed.
- N: The information is disclosed, but cannot be associated with a particular transaction. For example, this symbol would be used if the billing server knows a given consumer spent \$5 and a given merchant received \$5, but cannot be sure the two events are related.
- R: The information is disclosed to be within a given range, but the exact value is not disclosed.
- L: The information may not be disclosed, but must be when a valid warrant is presented by a law enforcement agency.
- ?: It is not known whether a transaction actually occurred.

Adjacent symbols with no intervening punctuation represent combined disclosure types. For example, "RN" means that the value is disclosed to be within a given range but cannot be associated with other values for the transaction. Symbols separated by a comma represent alternatives. For example, "N, L" means that the value is

disclosed but cannot be associated with other values, yet can be fully disclosed to law enforcement agencies with a warrant.

2.3. Properties of the Matrix

The matrix described above exhibits a number of interesting properties. They are:

- Anything disclosed to an observer is known by all other parties.
- Each of the participants has complete knowledge of his own identity, address, and account, and complete knowledge of the items, amount, and tax status of the transaction.
- Detail information (such as account numbers) may easily be hidden from observers using standard cryptographic techniques.
- Network addresses of participants may be difficult to hide; to achieve “R?” disclosure, we may use Chaum’s unconditional sender and recipient untraceability (see [4]), or forwarding-agent techniques (see section 4). Furthermore, we may use resale agents (see section 6.3.1) to dissociate participants from one another to achieve “RN?” status (the information is known to be within a given range for participants, but participants cannot be matched to one another or to a specific transaction).
- Given a network address or account number, it may be possible to determine a participant’s identity. This is not explicitly shown in the matrix; it should be understood that the columns corresponding to the consumer and merchant are related.

After considering these observations, the “most anonymous” transaction possible is similar to Table 2, with the exception of the *Authorities* row, as explained below. The *Auxiliaries* row is treated as the *Observer* row, reflecting the fact that the “most anonymous” transaction will not use auxiliary parties. We should always assume that the observer may be ideally placed to obtain the information (e.g., located on the same Ethernet segment and able to snoop packets), making the requirements for eliminating observable information as stringent as possible.

Table 2: Desired transaction information matrix.

	Consumer’s			Merchant’s			Items	Amount	Tax Status
	identity	account	address	identity	account	address			
Consumer	X	X	X			RN	X	X	X
Merchant			RN	X	X	X	X	X	X
Billing Server	N	N	RN	N	N	RN		N	
Authorities	L	L	RN?, L	L	L	RN?, L	L	L	L
Auxiliaries			RN?			RN?			
Observer			RN?			RN?			

2.4. Law Enforcement Access

For a system which has no explicit provision for disclosure to law enforcement, the *Authorities* row is identical to the *Billing Server* row, with “L” added to each entry. (In the table shown, law enforcement agencies with warrants have access to all information.) If we arrange our system to support only minimum disclosure, it is unlikely to be adopted, because of the potential for wire fraud or other illegal activities. Thus, we would like to provide the capability of revealing full information to law enforcement agencies.

Law enforcement agencies not possessing a warrant will be denied all information (except that available to any observer). We would like to provide law enforcement possessing a warrant with complete transaction information. Thus, the ideal matrix has with an “L” alternative added to every column in the *Authorities* row, as in Table 2.

3. NetBill Transactions

NetBill provides a funds transfer mechanism over the Internet. Consumers and merchants are authenticated using Kerberos (see [12]).

Consumers and merchants handle the initial parts of the transaction without intervention from the billing server, then request a transfer of funds from the billing server, which completes the transaction. The merchant acts as a liaison between the consumer and the billing server, thus simplifying the consumer's communications needs; messages between the consumer and billing server are encrypted using a Kerberos session key so that the merchant cannot eavesdrop, despite his liaison role. The arrangement of parties in the NetBill system is as shown in Figure 1. The numbers shown correspond to the steps in the explanation of the protocol in the next section. We are concerned only with the communications within the dotted-line box; communications between the NetBill transaction server and the bank are not considered.

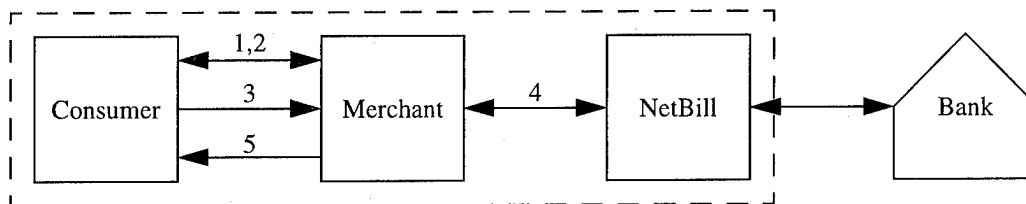


Figure 1: The arrangement of parties in the NetBill system.

3.1. Analysis of the NetBill Transaction Protocol

This section analyzes information disclosure in the NetBill system as described in [7] and [11]. Table 3 shows the final Transaction Information Matrix for NetBill transactions.

Table 3: Final Transaction Information Matrix for NetBill Transactions.

	Consumer's			Merchant's			Items	Amount	Tax Status
	identity	account	address	identity	account	address			
Consumer	X	X	X	X		X	X	X	X
Merchant	X	X	X	X	X	X	X	X	X
Billing Server	X	X	X	X	X	X	X	X	X
Authorities	L	L	X?, L	L	L	X?, L	L	L	L
Auxiliaries			X?			X?			
Observer			X?			X?			

Before the NetBill transaction begins, the consumer and merchant each know their own address, the consumer knows the merchant's address (this knowledge is assumed in order to begin communication), and the consumer knows which goods he would like to purchase, along with the purchase price and tax status.

1. *Consumer Requests a Service:* A consumer makes a request to a merchant indicating the intent to purchase an item for a specified price. In this phase, the consumer reveals his network address. Also, the merchant now knows what goods will be purchased. Eavesdroppers know the addresses of both parties, but not the amount and nature of the goods requested (which are encrypted). Because NetBill uses Kerberos authentication services, the merchant and consumer now also know each other's identities. Additionally, the consumer includes his account number in the request.
2. *Merchant Forwards Encrypted Goods:* The merchant encrypts the requested item with a key K and forwards the goods to the consumer. The merchant creates an electronic invoice, and calculates a hash of the encrypted goods, placing the result in its invoice. This phase does not affect the availability of transaction information.

3. *Consumer Acknowledges Receipt of Goods*: The consumer creates an electronic payment order (EPO), which is filled in with such pieces of information as its identity, account number, and its own calculated hash of the encrypted goods. The EPO is sent back to the merchant to acknowledge the receipt of the encrypted goods. Because the consumer's EPO is encrypted with a session key known only to the consumer and the billing server, this phase does not affect the availability of transaction information.
4. *Merchant Invokes NetBill Transaction*: The merchant passes its invoice and the consumer's EPO to NetBill. NetBill will examine the contents of both of these to determine whether the transaction is valid. Invoice and EPO components such as price, item, parties and hash are used in verification. The merchant's invoice also contains the decryption key K for NetBill to store. If the proposed transaction is deemed valid, NetBill will execute and log the transaction. The merchant is notified of the transaction results and passes this information on to the consumer. With this phase, NetBill knows all relevant information and, upon presentation of a warrant, will reveal it to law enforcement authorities. In addition, the consumer's monthly statement will include the account numbers of all merchants with whom he has done business.
5. *Merchant Supplies Key*: Upon notification of a successful transaction, the merchant returns the decryption key K for the delivered goods to the consumer. This key will be maintained at both the merchant and NetBill locations in case the consumer needs to re-request it. This phase does not affect the availability of transaction information.

Because electronic goods (such as documents or software) are delivered in encrypted form, they are unusable until payment is made; the key is transmitted as part of the funds transfer request and forwarded to the consumer upon successful completion, so it is impossible for the consumer to steal goods by aborting the transaction before payment. ([1] has a general discussion of this technique, known as *certified delivery*.)

4. Hiding Network Addresses

In distributed transactions, because of the need for the participating parties to exchange messages, it seems natural that the parties must know each other's address. This is not necessarily the case, however. In [4], Chaum describes a method of hiding addresses using broadcast messages and cooperating potential senders. This section introduces an alternative method for parties to mask their addresses using forwarding agents.

4.1. Forwarding Agents

We assume the existence of message forwarding agents, whose addresses are well known. In order that agents may know the final intended recipient, each participant must know some distinguishing characteristic of its peer. I call this piece of information the participant's *tag*. We assume that forwarding agents know how to get messages to parties based on their tags, either because they have access to a tag-to-address directory or because participants can poll tag dropoff points periodically (as a person might poll a Post Office box).

In the following analysis, I will use A and B to represent the participants. I use G and H to represent single forwarding agents, and F to represent the complete set of all available forwarding agents. M represents a message and T_A is a tag representing participant A . In all cases, A and B use end-to-end encryption so that none of their communication is revealed to the forwarding agents.

4.1.1. Basic Single Agent

In order to send a message to a peer B whose address is not known, a participant A sends the message M with attached tag T_B to a forwarding agent G selected from the set F of available agents. G replaces T_B with T_A (to indicate the sender of the message, as a return address) and forwards the message to B .

Is this secure? No, for several reasons. First, the agent G knows the addresses of both participants, and therefore must be trusted not to reveal this information. (Indeed, it may be that agents know the addresses of *all* potential participants. The important point here is that the agent knows that these two are communicating.) Second, to find its peer's address, either party may send a message and then eavesdrop on the agent's outgoing messages looking for one with its own tag as the return tag. Finally, a rogue participant may pose as the forwarding agent and convince the other to send messages directly to himself, thus revealing the victim's address.

The next section shows how to overcome these problems.

4.1.2. Encrypted Single Agent

This arrangement is similar to the previous one, except that in addition to the end-to-end encryption employed by *A* and *B*, each pair of parties encrypt their point-to-point communications with a different key. This may be accomplished easily with any of various key exchange protocols such as the scheme presented by Diffie and Hellman in [6], or even a Kerberos-based authentication step.

This shares the weakness of the previous arrangement that the forwarding agent must be trusted, but prevents the parties from eavesdropping on the agent's communications with other parties or (if the system provides authentication) posing as the agent. It may also seem that traffic analysis would still be possible; section 4.2 presents the relevant traffic analysis issues.

4.1.3. Encrypted Multiple Agent

We can eliminate the necessity of trusting forwarding agents by using two or more forwarding agents. No single agent knows more than one participant; only by collusion among all forwarding agents may both participants be discovered.

An originating party, *A*, selects two forwarding agents *G* and *H* from the set *F* of agents. *A* packages the message to *B* as if it were using only *H* to forward the message, and then treats that as a message to be sent to *H* through *G*. *G* does not know that the final recipient is *B*, and *H* does not know that the original sender was *A*. Again, messages are encrypted with a different key for each pair of communicating parties, as in the Encrypted Single Agent arrangement. This arrangement is similar to the use of "cascades" in [3].

This presents a trade-off between security and complexity: the more forwarding agents there are in a chain, the more difficult it is for an adversary to gain the cooperation of all of them, but the more complexity is involved in sending messages.

4.2. Foiling Traffic Analysis

An observer may try to obtain the network addresses of the participants in an electronic transaction using traffic analysis. This section presents three common techniques and explains how they may be defeated.

4.2.1. Message Content Correlation

In the first kind of attack, the observer watches packets being transmitted across a network, and compare the contents of those packets with each other. For example, if an observer wants to know with whom I am communicating, he can observe the packets coming from my workstation and follow those packets around the network (by watching for packets with the same content) until they reach their destination. This is the attack mentioned above, used to demonstrate the need for encrypted channels. As stated above, it is important that communications are first encrypted end-to-end, and then the encrypted messages are encrypted a second time with different keys on every point-to-point link.

4.2.2. Message Length Correlation

This second type of attack is similar to the first, in that the properties of a message are compared to track individual messages around the network. In this case, however, the length of the message is used, rather than its content (perhaps because the contents of messages are encrypted, as suggested above). Thus, an adversary may track unusually long or short packets around the network.

A simple way to ensure that packets cannot be tracked in this manner is to require all packets to be of a fixed length (by padding short messages and segmenting longer messages, for example). If all packets are the same length, an attacker has no way of determining which message emerging from a forwarding agent corresponds to which of the messages received by that agent.

4.2.3. Message Timing Correlation

If an adversary cannot rely on the content or length of messages to track them around the network, he must rely on the timing of messages to associate them with one another. When one workstation transmits a packet, the attacker can eliminate as possible recipients all the workstations on the network which do not receive packets within a "reasonable" time (the definition of which, naturally, depends on the nature and usage patterns of the network and other factors).

There are some fairly simple ways to reduce the effectiveness of this sort of attack; these may be used independently or in combination.

First, forwarding agents may introduce a random delay before forwarding any message, thus reducing the correlation between time of transmission and time of receipt. The main problem with this method, of course, is that the longer the introduced delay, the slower the communication between end parties becomes (especially if retransmission schemes are used to implement reliable transfer protocols). And if a small delay is used, much of the timing correlation is preserved. Selecting a delay value is a design trade-off between speed of communications and difficulty for the attacker.

Second, we may use distractor messages, introduced at various intervals; their recipients may detect and ignore these messages, while other parties in the system cannot distinguish them from legitimate messages. In addition, we may send messages along redundant paths, so that some of the messages may be dropped at random. If many senders send many messages along multiple paths, the task of tracking them through the network becomes extremely complex, because it is difficult to associate any given set of packets with a given source.

One example of this might be a system in which every participant transmits one packet during every fixed time interval; those without a useful packet to send would send a "noise" packet. If every participant is transmitting all the time, it becomes extremely difficult for an adversary to associate any packet with its source. If this method is combined with the previous method, the adversary's task becomes truly staggering: even for a single participant, it becomes impossible for an observer to be certain, after a single stage of forwarding agents, which packets contain legitimate messages and which do not.

5. Pseudonyms

It is often to our advantage not to be completely anonymous, but rather to be identifiable as a consistent party about whom full information is not available. The creators of many current anonymity schemes understand this, as is evidenced by the existence of the "Persona" PEM public key certification authority run by RSA Data Security, Inc. (see [9] for a description of the PEM public key certification hierarchy), as well as most anonymous-remailer systems, which assign each user a fixed pseudonym.

In NetBill, in addition to authenticating a consumer's right to spend on an account, a consumer's identity is used for two things: first, consumer-based pricing discounts, in which merchants may base their quoted prices on prearranged contracts or volume discounts; second, "blacklisting," where merchants may refuse service to a consumer based on past abuse (a history of disputed transactions, for example) or other factors.

However, we wish to prevent the sort of abuse that comes from using the same pseudonym with every merchant: merchants can correlate purchase records and build profiles of consumers without knowing their real identities (it may even be possible to deduce the real identities given enough information from enough merchants).

So, we need a pseudonym system that allows consumers to use a different pseudonym with each merchant. It should not, however, allow a consumer to use more than one pseudonym with a given merchant; that would allow a consumer to defeat blacklisting, or prevent merchants from offering consumer-based pricing schemes. And, as in any pseudonym system, it should not be possible to determine the consumer's identity given the pseudonym, either by direct mapping or by verification of repeated guesses.

If we form consumers' per-merchant pseudonyms by taking the secure digest of a combination of the consumer's identity and the merchant's identity, along with a secret bit sequence known only to the pseudonym generator, (using a secure message digest algorithm such as the MD5 algorithm described in [10], the Rabin-Karp algorithm described in [8]), we achieve the goals as stated.

With this system, a consumer wishing to connect anonymously to a merchant would request a Kerberos ticket from the Ticket Granting Server with the pseudonym as principal name. The consumer and merchant authenticate and communicate using the pseudonym only.

Because the secure digest algorithm cannot be reversed to reveal the consumer's identity, this system succeeds in preserving the anonymity of the consumer. Because the secure digest is deterministic and depends only on the consumer's and merchant's identities and a fixed secret key, a consumer cannot use more than one pseudonym with a given merchant. Finally, because of the low probability of collisions it is extremely unlikely that any two consumer-merchant combinations will have the same pseudonym.

6. Integration Into NetBill

Comparing the Transaction Information Matrix for the proposed NetBill protocol (Table 3) to the desired matrix (Table 2) reveals several areas for improvement. First, the consumer and merchant know each other's identities, from the use of Kerberos authentication between them. Second, the consumer and merchant know each other's network addresses, and external parties may associate pairs of communicating parties, because they communicate directly over TCP/IP. Third, the NetBill server knows all information in the transaction, which is revealed by the consumer's EPO and merchant's invoice. In this section, I apply techniques discussed earlier in the paper to these problems, improving privacy properties of NetBill.

6.1. Eliminating Identity Information

By allowing consumers and merchants to use pseudonyms to authenticate to one another, we may protect their privacy while retaining the desirable features of the Kerberos model (strong authentication and establishment of a shared session key for communication) and of a consistent identity for a consumer-merchant pair (ability to implement special pricing schemes and blacklisting).

6.2. Hiding Participants' Addresses

If we use the encrypted multiple forwarding agent scheme between the consumer and merchant, we can reduce knowledge of the consumer's network address to "RN" for all involved parties and "RN?" for all uninvolved parties. Additionally, we reduce knowledge of the merchant's network address to "RN" for the consumer and "RN?" for uninvolved parties (it remains "X" for the NetBill server). Furthermore, if we use the encrypted multiple forwarding agent scheme between the merchant and NetBill, the merchant's address becomes as well-hidden as the consumer's ("RN" for involved parties, "RN?" for uninvolved parties).

It seems that the additional network bandwidth required to use noise messages and redundant paths for this purpose is too costly to be worth the additional benefit it provides, but they may be desirable for applications requiring additional levels of security.

6.3. Minimizing Centralized Information

The billing server has complete knowledge of the transaction, including participants' identities and account numbers, items purchased, amounts and tax status. In a digital cash system, the messages exchanged themselves represent negotiable value, and may pass through many hands before being exchanged for actual currency, as described in [2]. Because NetBill is a funds-transfer system, however, it is very difficult to hide participants' identities and transaction amounts from the server.

Hiding the nature of the items and tax status is very simple on the surface: we can simply remove those fields from the invoice and payment order that the consumer and merchant send to NetBill in their transaction request. However, NetBill uses the item catalog number to verify that the consumer and merchant agree on the item to be sold; the item description and tax status are as simple for NetBill to obtain as they are for a potential consumer. Additionally, it is convenient for consumers to have item catalog numbers on their NetBill statements at the end of a billing period; it would not be acceptable to simply provide a list of amounts.

Nonetheless, it may still be possible to hide this information from NetBill by one of the following methods.

First, the consumer and merchant could simply agree not to tell NetBill what the items were. All NetBill verifies is that the consumer and merchant agree on the item that is to be sold—that is, they both include the same product number in their invoice or payment order. If they both agree to give the item a generic product number (for something that is listed as, for example, "General Merchandise"), NetBill will approve the transaction and still have no way of knowing what goods or services were actually transferred.

Second, the consumer may use a *resale agent* to hide many pieces of information from other parties.

6.3.1. Resale Agents

Resale agents can hide many pieces of information from many parties. In fact, they can be used in place of other techniques described in this paper to hide some pieces of transaction information. For example, it is possible for a consumer to hide his address to some extent from a merchant by using a resale agent. They are somewhat less effective, however, because resale must be trusted by consumers, and because they introduce several obstacles to convenient purchases.

To use a resale agent, a consumer prepares a list of goods or services he wishes to purchase, from various merchants at various prices. He transmits this list to the resale agent, who individually purchases the items from the merchants and resells them in one lump transaction to the consumer.

Clearly, the consumer cannot be associated with the merchants, except through the resale agent. (It is assumed that enough people use resale agents that the mere fact a given consumer is using a given agent is not sufficient information to link the consumer with the purchases.) If the transaction between the consumer and the resale agent is a sufficiently large aggregate transaction, the transaction price will not be enough information to link the consumer with individual purchases. If consumers, resale agents and merchants use anonymous forwarding agents to communicate, the NetBill transaction matrix becomes Table 2 with no further modifications.

One difficulty with this arrangement is that the consumer must trust the resale agent not to give information away. If the resale agent is corrupt, he can give away any or all of the consumer's sensitive information, requiring no collusion with any other party. This problem may be addressed somewhat using techniques described earlier for hiding information from merchants. However, knowledge of the specific items is still vulnerable. Although this information may also be disclosed by the merchant, it is expected that a resale agent (who is a fourth party to the transaction) has less interest in maintaining the privacy of the transaction. (In fact, it is possible to use multiple stages of resale agents in a manner similar to the use of multiple forwarding agents, as described in section 4.1.3, in order to hide this information from individual resale agents.)

Another trust issue between the consumer and a resale agent arises because of the order in which the transactions take place. In purchasing through a resale agent, we have two alternatives.

In the first, the resale agent purchases all the requested goods before the consumer sends him payment. In this arrangement, the resale agent must trust the consumer to pay for the requested goods. In the second, the consumer sends payment for the goods before the agent purchases them. In this arrangement, the consumer must trust the agent to purchase and deliver the requested goods (for which payment has already been rendered; we cannot use certified delivery for this, because at the time payment was made, the resale agent was not in possession of the goods).

An alternative approach might be to use nested transactions. An advantage of this approach is that it would be possible to tie the results of transaction between the consumer and resale agent to the result of the transaction between the resale agent and the merchant, making it impossible to separate them. A disadvantage, however, is that the implementation of this type of transactions requires that the nested transactions are associated with one another in the logs, defeating the purpose of the resale agent.

It is worth noting that the use of resale agents without nested transactions requires no modifications to the NetBill model; it requires only merchants who act as resale agents and consumers willing to use them. The choice between the above policies belongs to the agent; consumers may exercise their preferences by choosing from among resale agents using their preferred policy.

6.4. Analysis of the Revised NetBill Model

Using the techniques described in this paper, it is possible to modify the NetBill transaction model (shown in Table 3) to more closely match the ideal disclosure shown in Table 2.

With these modifications, the billing server now has "N" for all entries, due to the use of resale agents to dissociate consumers and merchants. The merchant no longer has the consumer's account number, simply by omitting it from the negotiation. The merchant no longer has the consumer's identity, due to the use of pseudonyms. The transaction uses message forwarding agents, which removes the consumer and merchant's knowledge of each other's network addresses, and puts "N?" in the associated columns of the *Auxiliaries* row. The *Authorities* row still has "L" access to all information, though law enforcement authorities will now have to present a warrant to auxiliary agents as well as the billing server to obtain the full information. (Although each individual agent has only "N?" disclosure, all agents could cooperate to give them "X?" disclosure, thus reassociating parties with one another.) The completed table is shown in Table 4.

Table 4: Final Transaction Information Matrix for NetBill transactions with privacy enhancements.

	Consumer's			Merchant's			Items	Amount	Tax Status
	identity	account	address	identity	account	address			
Consumer	X	X	X	X		RN	X	X	X
Merchant			RN	X	X	X	X	X	X
Billing Server	N	N	N	N	N	N	N	N	N
Authorities	L	L	RN?, L	L	L	RN?, L	L	L	L
Auxiliaries			N?			N?			
Observer			RN?			RN?			

The NetBill system is currently undergoing design revision for a trial in 1995. [5] is a detailed exposition of the security choices made for the latest revision; it includes some of the techniques outlined in this paper.

Acknowledgements

There are several people without whom this paper could not have produced. I would d like to thank the NetBill project group for cooperation regarding the design and implementation of NetBill. My reader, Marvin Sirbu, was very helpful in discussions of many of the techniques described here. Finally, I'd like to extend my gratitude to my thesis advisor, Doug Tygar, for advice and direction on the research as a whole, and for detailed technical discussions on the issues presented here.

References

- [1] Alireza Bahreman and Doug Tygar. Certified electronic mail. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 3–19, San Diego, CA, February 1994.
- [2] DigiCash bv. *Digicash: Numbers that are money*. Product brochure. For information on DigiCash, send email to info@digicash.nl.
- [3] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
- [4] David Chaum. Security without identification: Transaction systems to make Big Brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
- [5] Benjamin Cox, Doug Tygar and Marvin Sirbu. NetBill Security and Transaction Protocol. *USENIX Workshop on Electronic Commerce*, July 1995.
- [6] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
- [7] Vaishali Goradia, David Lowe, David McNeil, Alexander Somogyi, and Thomas Wagner. *NetBill preliminary design*. Information Networking Institute internal report, June 1994.
- [8] Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, March 1987.
- [9] S. Kent. *RFC 1422: Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*. Internet Activities Board, February 1993.
- [10] R.L. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm*. Internet Activities Board, April 1992.
- [11] Marvin Sirbu and Doug Tygar. NetBill: An internet commerce system optimized for network delivered services. In *Proceedings of the 1995 IEEE Computer Communications Conference*, March 1994.
- [12] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter Conference*, pages 191–202, February 1988.

A Software Architecture to Support Misuse Intrusion Detection.*

Sandeep Kumar

Eugene H. Spafford

The *COAST* Project
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398
{kumar,spaf}@cs.purdue.edu

Keywords: intrusion detection, misuse, anomaly.

June 16, 1995

Abstract

Misuse intrusion detection has traditionally been understood in the literature as the detection of specific, precisely representable techniques of computer system abuse. Pattern matching is well disposed to the representation and detection of such abuse. Each specific method of abuse can be represented as a pattern and several such patterns can be matched simultaneously against the audit logs generated by the operating system kernel. Using relatively high level patterns to specify computer system abuse relieves the pattern writer from having to understand and encode the intricacies of pattern matching into a misuse detector. Patterns represent a declarative way of specifying what needs to be detected, instead of specifying how it should be detected. We have devised a model of matching based on Colored Petri Nets specifically targeted for misuse intrusion detection. In this paper we present a software architecture for structuring a pattern matching solution to misuse intrusion detection. In the context of an object oriented language used for the prototype implementation we describe the abstract classes encapsulating generic functionality and the interrelationships between the classes.

1 Introduction

Intrusion detection is an important monitoring technique in computer security aimed at the detection of security breaches that cannot be easily prevented by access and information flow control techniques. These breaches can be a result of software bugs, failure of the authentication module, improper computer system administration, etc. Intrusion detection has historically been studied as two sub-topics: *anomaly detection* and *misuse detection*. Anomaly detection is based on the premise that many intrusions appear as anomalies on ordinary or specially devised computer system performance metrics such as I/O activity, CPU usage, etc. By maintaining profiles of these metrics for different subject classes, for example individual users, groups of users, or programs and monitoring for large variations on them, many intrusions can be detected. Misuse intrusion detection has traditionally been understood in the literature as the detection of specific, precisely representable techniques of computer system abuse. For example, the detection of the Internet worm attack by monitoring for its exploitation of the `fingerd` and `sendmail` bugs [Spa89] would fall under misuse detection.

Several approaches to misuse detection have been tried in the past. They include language based approaches to represent and detect intrusions, such as ASAX [HCMM92]; developing an

*This work was funded by the Division of INFOSEC Computer Science, Department of Defense.

API¹ for the same purpose, such as in Stalker [Sma95]; using expert systems to encode intrusions such as in MIDAS [SSHW88], Haystack [Sma88], and NIDX [BK88]; and high level state machines to encode and match signatures² such as STAT [PK92] and USTAT [Ilg92]. We proposed using a pattern matching approach to the representation and detection of intrusion signatures [KS94b]. This approach resulted from a study of a large number of common intrusions with the aim of representing them as patterns to be matched against the audit trail [KS94a]. The signatures were also classified into categories based on their theoretical tractability of detection [Kum95]. We consider the following to be advantages unique to our model of pattern representation and matching.

- Sequencing and other ordering constraints on events can be represented in a direct manner. Systems that use expert system rules to encode misuse activity specify ordering constraints by directly specifying temporal relationships between facts in rule antecedents. This makes the Rete match procedure [For82] of determining the eligible production rules for firing, inefficient. STAT [PK92] and USTAT [Ilg92] permit the specification of state transition diagrams to represent misuse activity but their transition events may be high level actions that need not correspond directly to system generated events. ASAX [HCMM92] is the closest to our approach but it is less declarative. In specifying patterns in their rule based language RUSSELL, one must explicitly encode the order of rules that are triggered at every step. While ASAX tends to be a mechanism for general purpose audit trail analysis, our effort is a combination of mechanism and policy. The features provided in our work are closely tied to the intrusion characteristics we are trying to detect.
- Our model provides for a fine grained specification of a successful match. The use of pattern invariants (to be explained later) allows the pattern writer to encode patterns that do not need to rely on primitives built into the matching procedure to manage the matching, for example to clean up partial matches once it is determined that they will never match. This frees the matching subsystem from having to provide a complete set of such primitives and, in the process, couple the semantics of pattern matching with the semantics of the primitives.

Our method also has the following benefits but these are not necessarily a consequence of our approach.

Portability. Intrusion signatures can be moved across sites without rewriting them to accommodate fine differences in each vendor's implementation of the audit trail. Because pattern specifications are declarative, a standardized representation of patterns enables them to be exchanged between users running variants of the same flavor of operating system, with syntactically differing audit trail formats.

Declarative Specification. Patterns representing intrusion signatures can be specified by defining what needs to be matched, not how it is matched. That is, the pattern is not encoded by the signature writer as code that explicitly performs the matching. This cleanly separates the matching from the specification of what needs to be matched.

In this paper we describe our implementation of the model that was presented in [KS94b]. We have used C++ [Str91] as the programming language for the implementation of the prototype. The prototype runs under the Solaris 2.3 operating system and uses the Sun BSM [Sun93] audit trail as its input to detect intrusions. The programming techniques and language features we have used for the implementation are applicable to other programming languages as well. Our implementation is directed at providing a set of integrated classes that can be used in an application program to implement a generic misuse intrusion detector. The implementation also suggests a possible way of structuring classes encapsulating generic functionality and the interrelationships between the classes to design any misuse detector. The paper also describes that structure.

¹Application Programming Interface, i.e., a set of library function calls employed for representing and detecting intrusions.

²We use the terms intrusion signature and intrusion pattern synonymously.

Our choice of the language was dictated by the free availability of quality implementations of C++, our familiarity with it and the linguistic support provided in it to write modular programs. The set of integrated classes we have developed can be programmed in many other object oriented languages as well because no properties specific to C++ have been assumed or used. We only exploit the language's encapsulation and data abstraction properties. We use the word *class* in a generic sense and the corresponding notion from many other languages can be substituted here.

2 Our Approach

The model of pattern representation and detection on which the implementation is based was described in [KS94b]. Briefly, each intrusion signature is represented as a specialized graph in this model. These graphs are an adaptation of Colored Petri Nets described by Jensen [Jen92] with guards defining the context in which signatures are considered matched. Vertices in the graph represent system states. The pattern represents the relationship among events and their context that forms the crux of a successful intrusion or its attempt. Patterns may have pre-conditions and post-actions associated with them. A pattern pre-condition is a logical expression that is evaluated at the time the pattern springs into existence. It can also be used to set up state that may be used later by the pattern. Post-actions are performed whenever the pattern is matched successfully. For example, it might be desirable to raise the audit level of a user if he fails a certain number of login attempts within a specified time duration. This can be expressed as a post-action. Patterns may also include invariants to specify that another pattern cannot appear in the input stream while it is being matched. If a pattern is regarded as a set of event sequences P that it matches, and an invariant is regarded as another set of event sequences I that it matches, then a pattern with an invariant specification corresponds to the set $P \wedge \bar{I}$. A pattern can have more than one invariant. That corresponds to $P \wedge \bar{I}_1 \wedge \dots \wedge \bar{I}_n$. Invariants are needed to specify cases when it is no longer useful to continue a pattern match. For example, a pattern that matches process startups and records all file accesses by the process may require an invariant that specifies that matching be discontinued once the process has exited. From the practical viewpoint of specifying intrusion patterns, invariants usually result in more efficient matching rather than adding functionality to the pattern specification.

As a concrete example of a pattern, consider the monitoring of Clarke-Wilson [CW89] integrity triples in a computer system using the system generated audit trail. Clarke-Wilson triples are devised to ensure the integrity of important data and specify that only authorized programs running as specific user ids are permitted to write to files whose integrity must be preserved. This is similar to the maintenance of the integrity of the password file on UNIX systems by allowing only some programs, like `chfn`³ to alter it.

One pattern that might be used for this purpose is formed by a sequence of two sub-signatures: (1) that matches the creation of a process and (2) that matches any process writing to a file. By appropriately specifying that the created process is the same as the one that writes, and retrieving the user id, the program name, and the file name from the context of the match, Clarke-Wilson integrity triples can be monitored. See figure 1 for a pictorial representation of the signature.

The implementation of this model can be broken down into the following sub-problems:

1. The external representation of signatures. That is, how does the signature writer encode signatures for use in matching.
2. The interface to the event source. In our example it would be the interface to the C2 audit trail.
3. Dispatching the events (audit records) to the signatures and the matching algorithms used for matching.

³`chfn` is used to change information about users which is stored in a well-known file, `/etc/passwd`.

A PROGRAM STARTS UP	A PROCESS WRITES TO A FILE
PR = this program's name	F = this file's name
PID = this process's pid	PID' = this process's pid

Context: $PID = PID' \wedge$ Clarke-Wilson access triples do not permit PR running as user id PID to write to file F.

Figure 1: Monitoring Clarke-Wilson triples as a pattern match.

These issues are discussed in the next section. In addition to solving these requirements, our implementation is designed to simplify the incorporation of the following:

- The ability to create signatures and to destroy them dynamically, as matching proceeds.
- The ability to partition and distribute signatures across different machines for improving performance.
- The ability to prioritize matching of some patterns over others.
- The ability to handle multiple event streams within the same detector without the need to coalesce the event streams into a single event stream.

We describe our design in the next section and show how the library classes implement the design.

3 Overall Architecture

The library consists of several classes, each encapsulating a logically different functionality. An application program that uses the library includes appropriate header files and links in the library.

The external representation of signatures (sub-problem 1) is done using a straightforward representation syntax that directly reflects the structure of their graph. These specifications can be stored in a file or maintained as program strings. When a signature is instantiated in an application, a library provided routine (a Server class member function) is called that compiles the signature description to generate code that realizes the signature. This code is then dynamically linked to the application program and pattern matching for that signature is initiated. The application also instantiates a server for each type of event stream used for matching. Events are totally encapsulated inside the server object (sub-problem 2) and are only used inside signature descriptions. As signature descriptions are compiled they are added to the server queue. The server accesses and dispatches events to the patterns on its queue in some policy specifiable order (sub-problem 3).

The application structure is explained below which gives an overall view of the application. Section 3.2 looks at the structure of events. Section 3.3 explains the structure of the server itself in detail and its relationship to the patterns that are instantiated by the application.

3.1 Application Structure

As an example application structure, consider matching the pattern described in figure 1. This may look as shown below.

```
//file application.C
1  #include "C2_Server.h"
2
3  int main()
4  {
5      C2_Server S;
6      C2_Pattern *p1 = S.parse_file("CW"); //read signature from "CW"
7
```

```

8      /* duplicate a thread of control if necessary. run() doesn't return */
9      S.run();
10
11     return(1);
12 }

```

The application program makes use of a `C2_Server` object. The server object understands the layout of events and the event types that can be legally used in a signature definition. `C2_Server` also knows how to access events, in this case from the audit trail, and how to dispatch them to the signatures that are registered with it. The server is also responsible for parsing signature descriptions and can check it for correctness because it understands the data format of the events. The call to the server member function `parse_file` reads, compiles, and registers a new pattern with the server object. When the server object member function `S.run()` is called, it starts reading events and dispatching them. This consumes one thread of control as `S.run()` never returns. The server is responsible for implementing concurrency control among its member functions to ensure that concurrent calls to its public member functions do not corrupt its internal state. Our implementation uses the idea of monitors [Hoa74] to ensure this. The pattern description contained in file `CW` looks as shown in listing 1 below. The pattern is written to match against the Sun BSM [Sun93] audit trail.

```
//file patterns-ip
```

```

1  pattern CW "Clarke Wilson Monitoring Triples" priority 10
2  int PID, EUID; /* token local variables. may be initialized. */
3  str PROG, FILE;

```

`PROG` is a token local variable that stores the program name corresponding to the process id `PID`, `FILE` stores the file name that `PROG` opens for writing. `EUID` stores the effective user id of `PROG`.

```

4  state start, after_exec, violation;
5  post_action {
6  printf("CW violated for file %s, PID %d, EUID %d\n", FILE, PID, EUID);
7  }

```

The post action is code that is executed when the pattern is successfully matched.

```

8  neg invariant first_inv
9  state start_inv, final;
10
11  trans exit(EXIT)
12  <- start_inv;
13  -> final;
14  |_ { PID = this[PID]; }
15  end exit;
16  end first_inv;

```

The invariant specifies the removal of partial matches once a process has exited. What follows is the pattern description. The pattern matches all `EXECVE` records to monitor the creation of all processes in the system. Once a process creation is matched, the pattern further attempts to match all possible ways in which the process could modify a file. These could be:

- Open a file to read and create it if it doesn't exist. Or, open a file to read and truncate it if it exists.....and so on for all the other valid audit record types involving an open that might change the file. These are handled in transition `mod1`.
- Delete a file. This is handled in transition `mod12`.

```

17  trans exec(EXECVE) /* EXECVE is the event type of the transition */

```

```

18     <- start;
19     -> after_exec;
20     |_ { this[ERR] = 0 && PID = this[PID] && PROG = this[PROG] &&
21         EUID = this[EUID]; }
22 end exec;
23
24 trans mod1(OPEN_RC|OPEN_RTC|OPEN_RT|OPEN_RW|OPEN_RWC|
25           OPEN_RWTC|OPEN_RWT|OPEN_W|OPEN_WC|OPEN_WTC|OPEN_WT)
26     <- after_exec;
27     -> violation;
28     |_ { this[ERR] = 0 && PID = this[PID] && FILE = this[OBJ] &&
29         disallowed(EUID, PROG, FILE); }
30 end mod1;
31
32 trans mod12(UNLINK)
33     <- after_exec;
34     -> violation;
35     |_ { this[ERR] = 0 && PID = this[PID] && FILE = this[OBJ] &&
36         disallowed(EUID, PROG, FILE); }
37 end mod12;
38 end CW;

```

Listing 1: A Sample Pattern Description

If an application needed to match patterns against IP datagrams, it might have used an IP_Server instead of C2_Server or concurrently with it within the same application program.

3.2 Event Structure

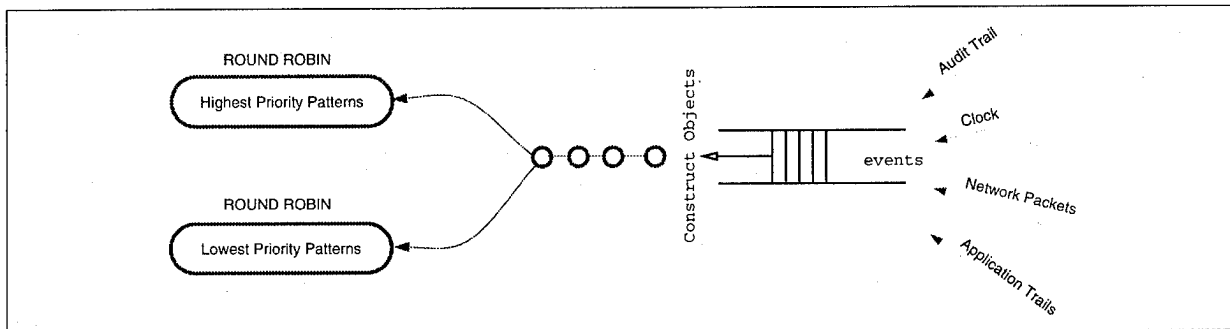
Each event in the event stream is converted to an instance of an event class. For handling a C2 audit trail this class might be named C2_Event. This class encapsulates all the attributes common to C2 audit records. Derived classes of C2_Event can be used for specifying more specialized types of audit records. For example, C2Event_EXECVE and C2Event_LINK can be derived to represent audit records generated by the execve and link system calls. Each event object can identify its type through its type() member function. This is used by the server to identify an event before dispatching it to the appropriate patterns. All the data belonging to the event is made available through its member functions. This encapsulates the organization of data in the event, which may be system dependent in general. The description of all the event classes constitutes the backend of the system and is one of the few system dependent layers.

3.3 Server Structure

For each event, the server looks at its type and consults a dynamically maintained table of patterns that have requested events of that type. It then calls the Patproc procedure of each such pattern. Patproc is a procedure associated with every pattern (its member function) that handles events for it. This approach to handling events is similar to the approach taken in Microsoft Windows [Pet92]. Events that are referenced in a signature description are explicitly requested by the signatures for dispatching when they are instantiated.

Events can be dispatched to patterns based on their priority. Patterns can be placed in queues at the appropriate priority level, and patterns serviced in each queue in a round robin fashion. This ordering of patterns by priority assumes that on the average, an event can be dispatched to all the patterns requesting it in a time less than the mean time of generation of an event. If this requirement is not met, patterns up to a certain level in priority may be perpetually starved. A

mechanism to age patterns in which patterns that have not been exercised by any event for a length of time have their priority increased, can be added. Pictorially this may look like:



Our prototype does not currently implement the priority structure of dispatching events to patterns. It treats every pattern to be of the same priority.

3.4 Summary

The use of an event stream with the detector requires the creation of two classes. One event class that is the root class of all events provided in the event stream; the other, a server class that parses pattern descriptions, instantiates them and manages them on its data structures. The server class interacts with the event class by converting raw events into objects of this class and dispatching them to patterns. The interrelationship between the various classes is shown in figure 2. It shows how two event streams, namely IP datagrams and C2 audit records can be used together. IP_EVENTS is the root class from which all the events corresponding to IP datagrams can be derived. Similarly, C2_EVENTS is the root class for deriving C2 audit record objects. Signatures written to match against IP datagrams are queued in an instance of IP_SERVER, while signatures that match against the audit trail are queued in an instance of C2_SERVER. Class names bounded by dotted boxes are abstract classes. The functions identified within these boxes are the pure virtual functions of these classes.

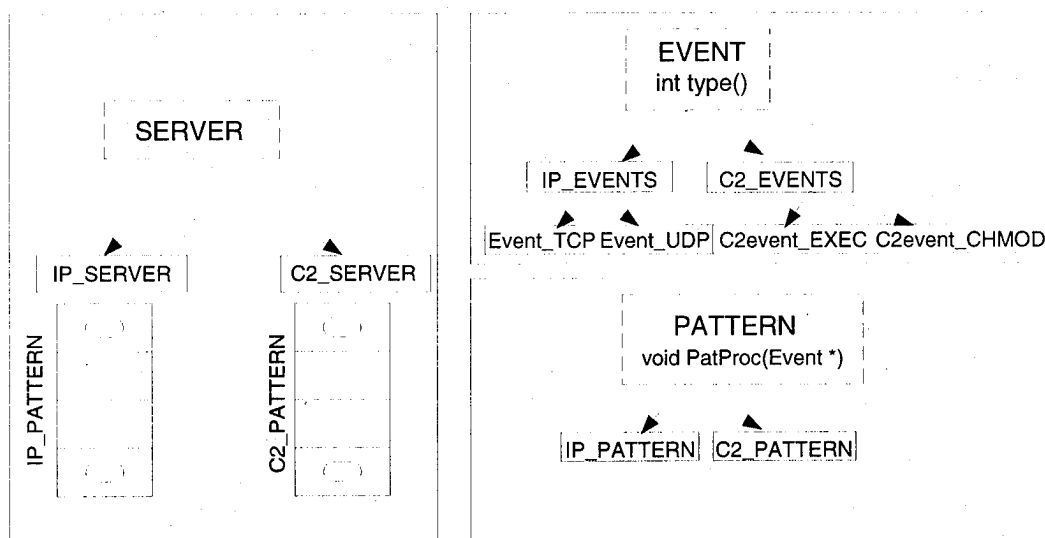


Figure 2: Interrelationship between the various classes in the detector.

4 Design Choices

By far the most significant consideration guiding the design was the run-time efficiency of the detector. For misuse detection using a C2 generated audit trail, one might reasonably expect

to process events (audit records) at the rate of 50K-500K/user/day [Sma95]. Furthermore, any computer resource required for matching signatures reduces the availability of these resources for general use. We therefore decided not to interpret the pattern automata by using table lookups to determine the pattern structure, but instead to compile the pattern description into an automaton. This also has the benefit of compile time optimizations of guard expressions present in the pattern. As the generated code realizing the automaton did not need to be "user friendly," we tried to make it more efficient by using functions as little as possible to avoid function call overhead in cases where functions could not be inlined. This often meant that data structures manipulated by the various pieces of the generated automaton were not encapsulated and were manipulated directly by these pieces. This has not proved to be a problem as the routines that generate this "program" are structured and the generated program logic can be deciphered by following the structure and logic of the generating routine.

The overriding constraint of efficiency combined with the requirement to dynamically create and destroy patterns meant that automaton descriptions be compiled and dynamically linked for the purpose of matching. An additional benefit of the dynamic creation of patterns is that new patterns can be created within an executing program based on the logic and execution flow of the program. For example, it might be desirable to instantiate specific patterns for matching based on the type and degree of observed suspicious activity. Such patterns may depend on the particular user and other specifics of the suspicious activity.

Our design, which is based on the model of dispatching events to patterns lends itself naturally for distribution. In a distributed design, the event sources (audit trails) may be generated on different machines and their processing on another machine. That is, the patterns, the server and the event sources may all reside on physically different machines. The server can then retrieve events by using any of several well known techniques [BN84, Par90] and dispatch them to patterns. Although our current implementation is single host based, a distributed implementation should be straightforward.

5 Performance

The experiments described below were done on a Sun SPARCstation 5 with 32MB of memory running Solaris 2.3 under light load. The audit file was generated separately by turning on auditing and simulating exploitations by hand and under program control. Auditing was configured with the default configuration which logs all events, both successful and failed. The pattern descriptions were translated into C++ code and compiled separately. The running times shown in the graphs below represent the reading of the audit file, conversion of each audit record into an object, and dispatching the event to all the patterns that request that event. It does *not* include the time for the matcher to load and begin execution, nor does it include the time to dynamically link the patterns.

Signatures were written for vulnerability data drawn from COPS [FS91], CERT advisories [CER] and the bugtraq and 8lgm⁴ electronic mailing lists.

Figure 1 shows how much time it took to match each signature against an audit file of approximate size 400KB⁵. Each sample point in the figure is the mean value of 200 runs. The small horizontal lines on either side of each point represents the standard deviation of the value over the runs. The audit file contained 2514 events. The sample point (0, 5.17) in the figure represents that the detector took 5.17s to create all the event objects and destroy them. The point (1, 5.45) means that pattern numbered 1 (numbered arbitrarily) took 5.45s when exercised by the 2514 events.

⁴Both lists discuss computer security vulnerabilities, their exploitation and steps for prevention and detection. Bugtraq is issued from bugtraq@crimelab.com and 8lgm advisories can be retrieved from fileserv@bagpuss.demon.co.uk.

⁵K in this section means 1000.

Some patterns take very little time, just a little over what it took to run with no patterns. The reason for this is that the type of events used in the pattern occurred so infrequently in the event stream that the cost of exercising the pattern on those events was negligible when compared with the creation and deletion of all the events in the audit trail. The mean time for the creation and deletion of an audit trail event is then $5.17/2514 = 2.1ms$. This is the fixed cost per event for the system.

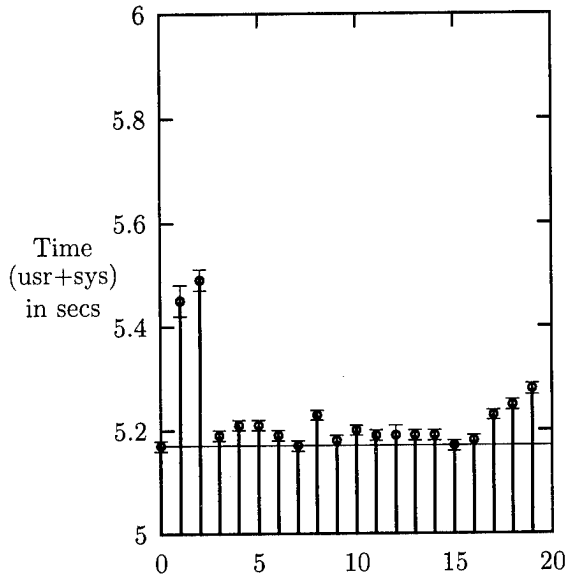


Figure 1: Time for matching each pattern for a 400K audit file.

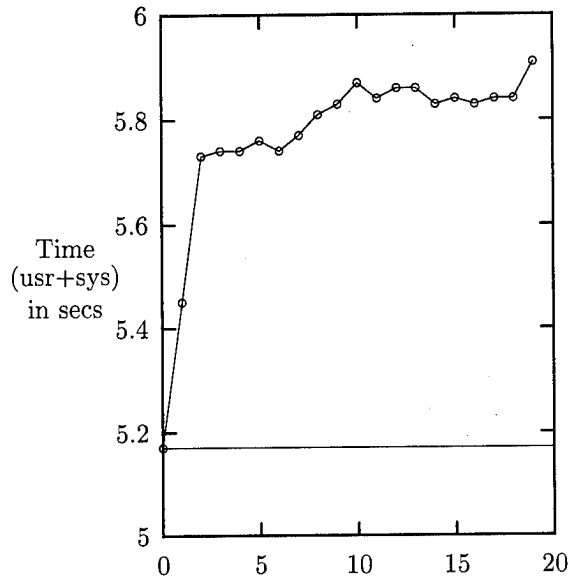


Figure 2: Time for matching multiple patterns for a 400K audit file.

Figure 2 shows the time taken when more than one pattern was matched simultaneously in the detector. The event stream and the pattern numbers are the same as in the previous simulation. In the figure, the data point (3, 5.74) shows that it took 5.74 to exercise the three patterns 1, 2, 3 together in the system. The fixed overhead cost of reading the audit file and converting each audit record into an object is the same as above, the varying cost that takes the multiplicity of patterns into account is:

$$\text{variable cost/event/pattern} \approx (5.91 - 5.17)/(2514 * 19) = 15\mu s$$

This calculation uses the data point (19, 5.91) which indicates that the detector took 5.91s to exercise 19 patterns together against an audit trail that consisted of 2514 events.

Consider the extrapolation of these results to estimate the performance of the detector in a more realistic setting. When running a set of programs in sequence that saturated the CPU, the Sun auditing subsystem generated about 1MB every 10 minutes on the single user SPARCstation. This translates to about 6MB per hour. This is about $2514 \times 6/.4 \approx 38K$ events per hour. Consider that there are 100 patterns in the detector. Then, for one hour of intense CPU activity, the detector requires the following time to process the generated audit data:

Fixed overhead	=	$5.17/2514 \times 38000s$	=	78.15s
Variable overhead	=	$15\mu s \times 100 \times 38000$	=	57s
Total time	=		=	135.15s

Thus, for every hour of intense activity, the detector requires $\approx 135s$ to match about 100 patterns. This fraction is $135/3600 \times 100 = 3.75\% \approx 4\%$ of the hourly activity. These results correspond to an unoptimized version of the detector.

6 Summary

This paper described a possible architecture for structuring a misuse intrusion detector based on pattern matching. The structure is client-server based in which the server obtains events and dispatches them to clients (patterns) which implement the matching procedure specific to their structure. Implementing this structure as a library permits embedding this type of matching within application programs. Our prototype allows the dynamic creation of patterns. These patterns are translated from a description language into C++ code that realizes the pattern and dynamically links that code into the application. The overhead of matching 100 signatures simultaneously against an audit trail that was generated at the rate of 6MB per hour on a Sun SPARCstation 5 was calculated to be under 5%.

7 Acknowledgements

We would like to thank all members of the COAST laboratory for their valuable comments on this paper, in particular Christoph Schuba for his extra effort and help with an earlier draft of this paper.

This work was supported, in part, by a gift from Sun Microsystems, and by DoD contract MDA904-93-C-4081: this support is gratefully acknowledged.

References

- [BK88] David S. Bauer and Michael E. Koblenz. NIDX - An Expert System for Real-Time Network Intrusion Detection. In *Proceedings - Computer Networking Symposium*, pages 98-106. IEEE, New York, NY, April 1988.
- [BN84] Andrew D. Birrell and Bruce Jay Nelson. Implementing Remote Procedure Calls. *ACM Transactions on Computer Systems*, 2(1):39-59, February 1984.
- [CER] CERT Advisories. Available by anonymous ftp from cert.sei.cmu.edu:/pub/cert_advisories.
- [CW89] David D. Clark and David A. Wilson. Evolution of a Model for Computer Integrity. *Report of the Invitational Workshop on Data Integrity*, September 1989.
- [For82] Charles L. Forgy. RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In *Artificial Intelligence*, volume 19. 1982.
- [FS91] Daniel Farmer and Eugene Spafford. The COPS Security Checker System. Technical Report CSD-TR-993, Purdue University, Department of Computer Sciences, September 1991.
- [HCMM92] Naji Habra, B. Le Charlier, A. Mounji, and I. Mathieu. ASAX: Software Architecture and Rule-based Language for Universal Audit Trail Analysis. In *Proceedings of ESORICS 92*, Toulouse, France, November 1992.
- [Hoa74] C. A. R. Hoare. Monitors: An Operating System Structuring Concept. *Communications of the ACM*, 17(10):549-557, 1974.
- [Ilg92] Koral Ilgun. USTAT: A Real-Time Intrusion Detection System for UNIX. Master's thesis, Computer Science Department, University of California, Santa Barbara, July 1992.
- [Jen92] Kurt Jensen. *Coloured Petri Nets - Basic Concepts I*. Springer Verlag, 1992.

- [KS94a] Sandeep Kumar and Eugene Spafford. A Taxonomy of Common Computer Security Vulnerabilities based on their Method of Detection. (unpublished), June 1994.
- [KS94b] Sandeep Kumar and Eugene H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11-21, October 1994.
- [Kum95] Sandeep Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, Department of Computer Sciences, (to appear) 1995.
- [Par90] Graham D. Parrington. Reliable Distributed Programming in C++: The Arjuna Approach. In *USENIX 1990 C++ Conference Proceedings*, pages 37-50, 1990.
- [Pet92] Charles Petzold. *Programming Windows 3.1*. Microsoft Press, 1992.
- [PK92] Phillip A. Porras and Richard A. Kemmerer. Penetration State Transition Analysis - A Rule-Based Intrusion Detection Approach. In *Eighth Annual Computer Security Applications Conference*, pages 220-229. IEEE Computer Society press, IEEE Computer Society press, November 30 - December 4 1992.
- [Sma88] Stephen E. Smaha. Haystack: An Intrusion Detection System. In *Fourth Aerospace Computer Security Applications Conference*, pages 37-44, Tracor Applied Science Inc., Austin, TX, Dec 1988.
- [Sma95] Steve Smaha. Talk given at the third Computer Misuse and Anomaly Detection Workshop (CMAD III) in Sonoma, CA, January 1995.
- [Spa89] Eugene Spafford. Crisis and Aftermath. *Communications of the ACM*, 32(6):678-687, June 1989.
- [SSHW88] M. Sebring, E. Shellhouse, M. Hanna, and R. Whitehurst. Expert Systems in Intrusion Detection: A Case Study. In *Proceedings of the 11th National Computer Security Conference*, October 1988.
- [Str91] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Publishing Company, 2nd edition, December 1991.
- [Sun93] SunSoft, 2550 Garcia Avenue, Mountain View, CA 94043. *Solaris SHIELD Basic Security Module Revision A*, October 1993. Part No: 801-5285-10.

PROVIDING ACCURATE DATA LABELS TO THE ANALYST THE SECURE C⁴I WORKSTATION

by Ingrid Dampier and Christine Corbett

TRW Integrated Engineering Division
One Federal Systems Park Drive
Fairfax, Virginia 22033-4411
Voice- (703) 803-4950; Fax: (703) 803-5096

Abstract

TRW has undertaken extensive research to define and prototype the intelligence analyst workstation of tomorrow. The objective of this research is to define an architecture that facilitates COTS and non development software (NDS) integration in a secure environment, and to validate the architecture in a prototype secure C⁴I workstation. The two overarching requirements for the prototype were to preserve the considerable investments already made in existing, fielded systems; and to demonstrate a system high mode of operations supported by trusted data labelling. The resultant secure reuse architecture is an open system architecture that integrates the existing fielded systems in a secure environment. We defined an architecture that uses a security isolation layer that provides security enforcing software, thereby minimizing the impact of security on the existing applications. Isolation of applications from security allows a system developer to take full advantage of commercial and corporate investment in application software, while still providing a secure operating environment that can be tailored to meet the specific requirements of the end user.

This paper explains the derivation of the workstation requirements, the system design approach, and the implementation considerations. The prototype uses the DIA-sponsored, compartmented-mode-workstation operating system, supplemented with trusted software to ensure accurate data labels are maintained, for both automatic and interactive processing. We conclude that the next generation analyst workstation can support state-of-the-art products *and* security.

Defining the Requirement

The first step in defining the next-generation C⁴I workstation was to determine the operational and security requirements of such a workstation. Operational requirements were readily derived from systems being developed and supported by TRW, including user recommendations for system enhancements. The primary capabilities required are automatic message processing, data fusion and database maintenance/interaction, office automation, and decision support and analysis tools tailored to the analyst's job. Numerous applications with these capabilities have been developed, fielded and tested in the operational environment. The research goal was to provide an architecture which supported the reuse of the applications best suited to the system mission, without incurring the expense of redesign and development. Early in our analysis, we reached the

conclusion that the investment already expended in fielded systems needs to be, and can be, respected and preserved. The C⁴I workstation of tomorrow needs to build on these systems and applications.

Security requirements however, offer a unique challenge because advances in security technology offer new system capabilities to protect sensitive information which can, and have, initiated a change in the requirements for the analyst workstation. The generation of products such as intelligence reports, database updates, and trend analysis represents the fusion of data from all sources. All data on the system must be available to support both manual analysis and automatic correlation decisions. In interactive analysis, the analyst must be able to see all the available data to make accurate deductions. At the same time, the analyst must make decisions on the resultant security classification of the product. These decisions require knowledge of the sensitivity level of the data sources.

Historically, intelligence systems are developed and operated in the system high mode. All users are cleared and briefed for all information on the system and all data is handled at system high. Therefore, intelligence products drafted by an analyst to be at a lesser sensitivity are defaulted to the system high classification and require downgrade procedures (often manual) to apply the appropriate classification before distribution. Manual downgrade can be a formidable task, if not supported by any maintenance of labels by the system. Without system provided labels, downgrading activities must rely on the analyst understanding the data context as it is presented, and some knowledge of the original sources. Given the increasing amounts of data being processed by intelligence systems, manual intervention in downgrading can seriously bottleneck the distribution of perishable data. Additionally, storage of classified materials incurs costs in management, control, and handling procedures. New workstation requirements call for a reduction in the amount of over classification so that the overhead of classified material handling and downgrading can be reduced. If accurate data labeling can be maintained, the over classification of data can be controlled.

The emerging requirement, therefore, is that systems maintain accurate classification of the data, while allowing the analyst access to all data for which there is a need to know. Through the appropriate application of evaluated trusted COTS operating systems and relational database management systems, the requirement to provide trusted labels can be met. The compartmented mode workstation (CMW) technology, sponsored by the DIA, provides a B1 compartmented mode, trusted, X-windowed environment on a security enhanced UNIX operating system. The CMW separates data based on classification label, including hierarchical level, compartments and code words, releasibilities, and handling instructions. The CMW encodings database provides the definition of rules, defined by the site, that control the aggregation of these labels, as well as the human readable format of the labels.

The technological challenge of our research was to combine the new trusted CMW operating environments with the current operational requirements. Existing, fielded analyst applications for the most part do not acknowledge separation of data based on classification; all data is treated at system high. These legacy applications are "label ignorant." The question to be answered was what type of architecture and supporting software tools were required to integrate these single-

level applications into an environment that did recognize accurate classification of data, without redeveloping the applications.

System Design Approach

We developed a cost-effective method of implementing new systems through integration (i.e., reuse) of existing applications. The software reuse architecture shown in Figure 1 stresses openness and expandability, allowing for replacement of applications with new applications, as technology extends the capabilities offered to analysts through automated data processing. The architecture provides isolation layers that insulate the application layer from the security policy and database access implementations. These isolation layers protect the from product changes in the data management and operating system layers. The data access isolation layer provides the necessary integration software that allows applications to be “plugged” into the architecture by providing data access translation to the databases. The security isolation layer (SIL) provides the security mechanisms not found in the COTS/NDS applications, such as data labeling, and interaction with the security mechanisms provided by the trusted CMW operating system.

The SIL is a set of trusted processes that implement the requirement to maintain the accurate classification of the data. The SIL uses the CMW operating system security mechanisms as a basis for its implementation. Trusted COTS database management systems, such as Trusted ORACLE 7 and Secure SYBASE are being evaluated as candidate RDBMS for the recording of the data classification labels with the data.

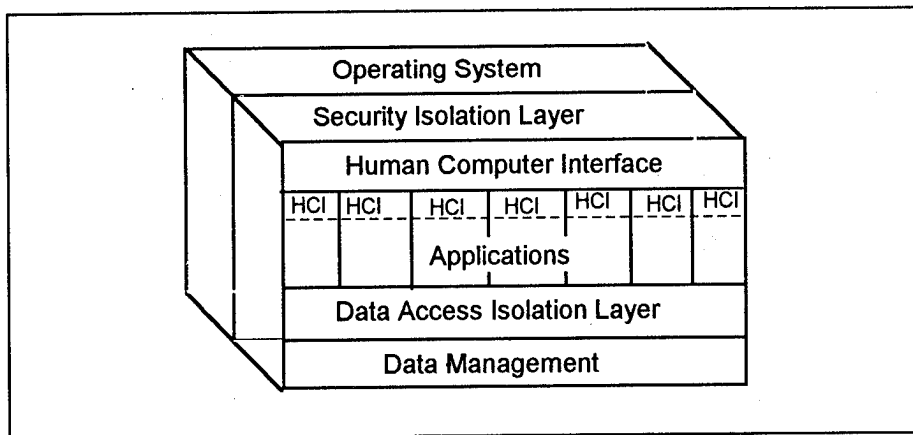


Figure 1. Reuse Architecture

The CMW provides two labels: the sensitivity label (SL), and the information label (IL). The SL is a static label that represents the maximum classification which a process or file can contain. The IL represents the maximum sensitivity of the data contained within the process or file. These labels are assigned by the CMW OS to the processes and to the data files. As data enters a process, the process IL “floats” up to the level of the data file or process providing the data. As additional data is read, the process IL continues to float so that it remains at the highest security

classification of all the data within the processing memory. Any data created by the process (i.e., written to disk) will be at the SL/IL of the process.

The CMW also provides for configurable privileges. Privileges are operating system rights to bypass the operating system security policy. If an application requires privileges, it needs to be "trusted" to not abuse that privilege. Such an application therefore needs to be designed, evaluated and tested specifically to ensure that it does not violate the security policy. The optimum secure reuse architecture would not require application software to run with privileges, thereby removing the need for applications to be trusted. The SIL provides for the central implementation of the security policy and manages the application processes so that they are not responsible for security-relevant decisions.

For example, the application creates a database record based on the parsing of incoming data (such as formatted text messages). The SL/IL of the resultant data file (or row in a RDBMS) would be set by the SL/IL of the parsing application. Existing message parsing processes, however, are "label ignorant." Therefore, the SIL determines a priori what the classification of the new record should be, based on the classification of the source message, and initializes the application at the appropriate SL/IL classification. Therefore, any data written by the application will be at the appropriate SL/IL, as managed by the operating system. The parsing application itself does not set or change the SL or IL. Our research shows that applications integrated in this manner require limited, if any, privileges.

Implementation Considerations

Because there is a spectrum of analyst capabilities required by a workstation, it is to be expected that the solutions also form a spectrum of complexity and cost. We developed a number of implementation designs that support the reuse architecture. To determine which of the possible SIL implementations should be used, we determined a criteria set to support a tradeoff analysis.

Use of Evaluated Products. Use of trusted products, particularly products that have been evaluated by NCSC, have a two-fold advantage. First, the products provide security mechanisms to support the application in meeting the federal criteria for trusted systems. These COTS-provided mechanisms, (such as object reuse, auditing, account management, and label management) reduce the amount of trusted applications software to be developed. Secondly, the evaluation and accreditation of systems using evaluated COTS products will be aided and expedited because a majority of the trusted computing base has been evaluated. Therefore, system accreditation of the final integrated system will be centered on assuring that the applications are correctly using the evaluated security mechanisms, without introducing new security vulnerabilities or risk.

Modifications to the existing applications. Modifications to modern NDS applications for which there is existing maintenance expertise available, including source code and current documentation, are generally not difficult, or expensive, to make. However, not all NDS have current expertise available, or current documentation. Therefore, it is important that the modifications required, if any are minor. The implementations supporting the secure reuse

architecture are limited to the modifications normally expected when porting to a new UNIX operating system.

Modifications to tailor COTS applications (trusted and untrusted) to the security architecture should be avoided in a viable implementation choice. While vendors may be willing to make modifications (for a cost), consideration must be given to supporting that modification. If the requested modification is deemed commercially attractive by the vendor, it may become part of the supported COTS product. However, it is rare all these conditions would be met within the schedule of the system being integrated. Therefore, our implementations assume that COTS will not be modified. All security relevant actions not provided by COTS will be handled in the SIL and not by adding functionality to the COTS applications.

Interactive Processing. Much of the interactive environment of an analyst workstation can be provided by untrusted COTS (e.g., office automation, geographic interaction tools, and multimedia capabilities). Our SIL implementations integrate these COTS as single-level applications that are not responsible for enforcing security. The SIL will interact with the trusted computing base for the COTS, so that the COTS product would not require privileges or trust. For example, our prototype encapsulates the COTS office automation (OA) product in an environment that sets the process SL/IL based on input from the user on creation and save of the OA files. When the analyst opens a new document, the SIL presents a template for the user to specify the security level of the document to be created. The SIL then initiates the OA at that level. Likewise, when the analyst opens an existing file, the SIL initiates the OA at the SL/IL of that file.

When entering data into the document, the OS ensures the IL of the document floats up to the highest security level of all the data being entered. This is determined by the SL/IL of the documents from which data is moved with a cut and paste operation, or by the SL/IL of the keyboard. The CMW OS has a mechanism by which the user can assign an SL/IL to the data being entered via keyboard or pointer.

On save, the SIL again present a template for the user to specify the correct classification of the data. Although the operating system, using the rules specified in the encodings file, aggregates the data labels from source documents, the document can be more or less sensitive based on the aggregation of the data content, or that the data selected for cut and paste was itself of lesser sensitivity than the entire originating file. The SIL can also be used to implement sanitization, either automatic (such as dirty word checks), or manual, including two person downgrade operations.

Automatic Processing. Automatic processing, such as message parsing and distribution, is vital to the analyst workstation. For automatic processing, the SIL implementation must provide for the maintenance of the data labels. Similar to COTS integration, we assume that for the most part, the NDS does not acknowledge data labels. We have several different SIL implementations to support NDS integration. The application must be examined as to the type of processing it performs, what data is used and what data is created.

First, the SIL can manage the labels at the point of entry/exit from untrusted applications. For example, the parsing process reads a message from a labeled source. The message file (or row in a RDBMS) has an SL/IL, previously applied by a trusted process that determined the message classification, either based on the origin (or communications channel) of the input message, or on message header/contents (embedded ASCII). Therefore, the SIL can initiate the parsing process at this SL/IL. For applications where the SIL can determine a priori the SL/IL of the output, this implementation is the most straightforward. We are working with several implementations of this design, optimizing how and when the processes are initialized. Each implementation is designed to work with the CMW OS mechanisms, to maximize the benefit of using a evaluated B1 product.

When the SIL cannot determine the SL/IL a priori (e.g., a correlation process), this implementation can still be applied with some modification. The IL float does not always accurately label data. For example, in the case of a correlation process, some data read by the process may be rejected, i.e. this data cannot be correlated with the current object. The discarded data may have caused the process IL to float in excess of the actual result. In this case, the SIL monitors the data as it is read by the process. The process returns information on which data actually were used in the creation of a new (or modified) data item. The SIL then re-initializes the process with only the relevant subset of data, allowing the SL/IL to be set by the OS process at the highest security level of these data items.

Summary

Our research and prototyping efforts validate that secure systems can be developed cost effectively. By reusing existing field proven applications coupled with an innovative integration architecture, we can demonstrate that accurate data labels are maintained, for both automatic and interactive processing. Our research continues to quantify the cost and document the methodology. The next generation analyst workstation can support state of the art products *and* security.

CONTROLLING NETWORK COMMUNICATION WITH DOMAIN AND TYPE ENFORCEMENT¹

David L. Sherman

Daniel F. Sterne

Lee Badger

Sandra L. Murphy

Kenneth M. Walker

Sheila A. Haghghat

Trusted Information Systems, Inc.
3060 Washington Road
Glenwood, Maryland 21738

Abstract

Today's operating systems cannot adequately control the processing of sensitive information in a network environment. Widely used systems lack mechanisms strong enough to enforce organizationwide restrictions on accessing sensitive information via network services. Mandatory access controls in trusted systems provide strength but lack flexibility needed to support site-specific integrity and role-based security policies. We describe how Domain and Type Enforcement (DTE), an operating system mechanism providing both strength and flexibility, has been extended and integrated with network services in a UNIX-based prototype. The approach provides uniformity across protocols, backward compatibility, and interoperability with existing IP networks.

1 Introduction

Today's operating systems cannot adequately control the processing of sensitive information in a network environment. One reason is that widely used operating systems like UNIX² generally lack access control mechanisms strong enough to enforce organizationwide restrictions on accessing sensitive information, especially via network services. Instead, these mechanisms allow authorized users great latitude to exchange information with other users, including users who are not equally authorized. As a result, these mechanisms are conducive to accidental misuse and vulnerable to manipulation by malicious programs. Mandatory access control (MAC) mechanisms in trusted systems provide stronger protection [14, 3], but are viewed by many organizations as inflexible or ill-suited for enforcing integrity and role-based security policies [6, 11]. Type enforcement [5, 20, 7, 12] is an access control mechanism that can provide both the strength and flexibility needed to support these kinds of policies [15, 22]. Research on type enforcement published to date, however, has focused on using it to protect information within a single isolated system; using it to secure networked systems has remained an open research issue.

This paper describes how Domain and Type Enforcement (DTE), an enhanced version of type enforcement, has been integrated with local area network (LAN) communication facilities in a UNIX-based research prototype. The prototype can enforce organizationwide restrictions on access to information in networked computer systems. These restrictions can be flexibly tailored to enforce integrity and role-based security policies and, more generally, to support the principle of least

¹Funded by ARPA contract DABT63-92-C-0020 - Approved for Public Release - Distribution Unlimited.

²UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

privilege [19]. This effort is part of a research project described by Badger, et al [1, 13], to make DTE practical and useful for near-term systems. The design goals for this effort include (a) conceptual and implementation compatibility with DTE facilities that control access to file system objects, (b) uniform protection across multiple transport protocols, (c) backward compatibility with existing UNIX binaries that use network services, and (d) interoperability with existing IP-based LANs. We do not discuss protecting information in transit over a network because the associated issues are separable and cryptographic techniques for addressing them are relatively well understood [9].

This paper is organized as follows: Section 2 provides background on DTE. Section 3 describes the conceptual model underlying our approach. Section 4 discusses our prototype. Section 5 is a summary.

2 Domain and Type Enforcement

DTE [1] is an enhanced version of an access control scheme proposed originally by Boebert and Kain [5]. In this scheme, an invariant attribute called a *domain* is associated with each subject; another invariant attribute called a *type* is associated with each object. Subject-to-object mediation decisions are made by comparing the subject's domain, the object's type, and the requested mode of access to a table [5] or database [1, 21] that describes the site's access control rules.

Like DoD MAC, DTE provides strong, organizationwide access control because the attributes of subjects and objects cannot be modified and because domains can be configured so that subjects have little or no ability to choose the attributes of objects they create. Moreover, DTE access control rules are protected from being modified by ordinary users and programs. DTE, however, is more flexible than DoD MAC. DTE access control rules can be configured by a system architect or security administrator to restrict access for a variety of purposes including least privilege [5], reliability, and safety [18]. In particular, DTE can be configured to support site-specific integrity and role-based policies [15, 22].

3 Conceptual Model

Our model for network controls encompasses three primary concepts: an interpretation of the principle of least privilege [19] as applied to a network environment, mediation, and network object abstractions.

3.1 Least Privilege in a Network Environment

To fully exploit DTE benefits, we envision DTE being used to confine each process to a domain in which it has access only to those objects that are essential to its assigned function and has only the required modes of access to those objects. In general, this implies that most systems will consist of many different types and domains and will require many cross-domain interactions. Some domains will provide access to a single type while others will provide access to multiple types. For brevity throughout this section, we will describe the use of DTE in this manner as supporting the principle of least privilege. It should be clear, however, that DTE can be used in this same manner to enforce integrity and role-based policies and address other engineering objectives.

In a single-host system, supporting least privilege means, in part, preventing unnecessary access to files, memory segments, and other locally stored objects. Consider two unrelated processes that need to obtain information from a file produced by a third process, as shown in Figure 1A. Only the

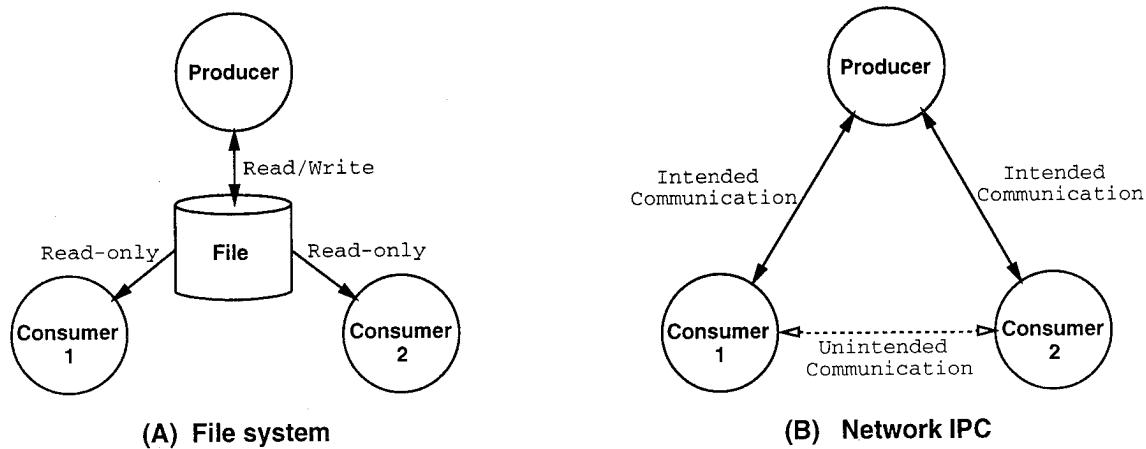


Figure 1: Least privilege as applied to process interactions

producer needs write access to the file, while each of the consumers needs read-only access to the file. Since the consumers are unrelated, neither needs the ability to write a file that the other can read. Least privilege then implies being able to prevent the consumers from using the file system to pass information directly to each other.

In a distributed system, the producer and the consumers might reside on different hosts and interact via network-based interprocess communication (IPC), as shown in Figure 1B. To support least privilege in this context, we need a means of allowing the consumers to communicate with the producer while preserving the ability to prevent the consumers from communicating directly with each other.

We would like to allow an architect or administrator to selectively permit or deny IPC between pairs of processes. As shown in Figure 2A, this can be accomplished by assigning processes to different domains and assigning appropriate data types to the objects they access via network IPC. In this figure the producer is in a domain that is distinct from each of the consumer domains. We assume that the domains of consumers 1 and 2 are different from each other, since the consumers are unrelated. The system architect can assign one type, *Request*, to the data that can be sent by both of the consumer domains and received by the producer's domain, and another type, *Response*, to the data that can be sent by the producer's domain and received by each of the consumer's domains. If there is no single type that can be sent by one consumer and received by the other, the consumers cannot directly communicate with each other.

Figure 2B shows a related example in which a server provides storage and retrieval services for multiple clients, some of which only consume while others both produce and consume. Both kinds of clients need to communicate bi-directionally with the server to use its services. Nevertheless, only the producer/consumer clients should be able to update the information stored by the server; pure consumers should not. Least privilege here implies being able to restrict the kinds of service requests that clients can submit to servers. As indicated in Figure 2B, this can be accomplished by associating types with different kinds of service requests and selectively incorporating into different

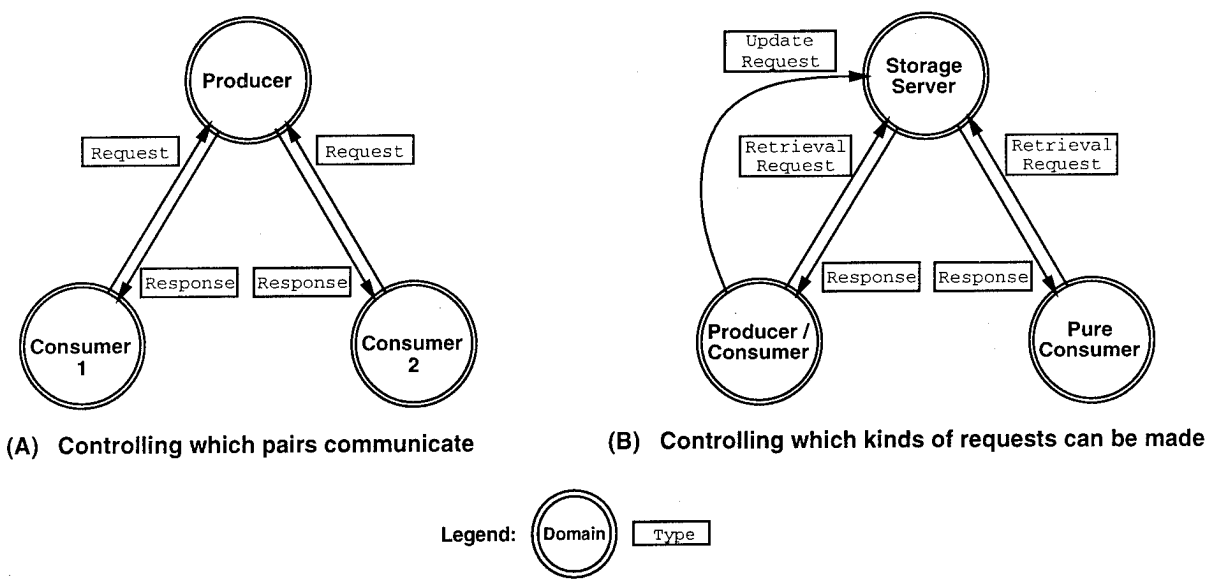


Figure 2: Controlling IPC with DTE

client domains the right to send information of those types. This ensures that service requests can be sent only by authorized clients and obviates the need for each server to authenticate and check the authorization of its clients.

In summary, supporting least privilege in a network environment implies being able to control which pairs of processes can communicate via IPC and which types of information each pair can exchange.

3.2 DTE Message Mediation

The primary IPC services requiring mediation are the sending and receiving system calls, illustrated in Figure 3. In our conceptual model, network IPC services manipulate objects called *typed messages*, which are described in the next section.

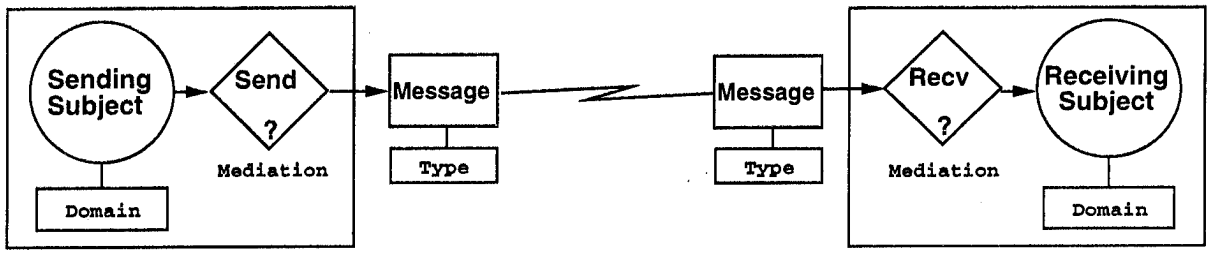


Figure 3: Mediation

messages, which are described in the next section. When a process sends data, it can send only one type of data per system call. The process may explicitly specify the data type; otherwise, the

DTE mechanism will automatically derive a default type from the DTE rule base [1]. The DTE mechanism determines whether or not the domain of the process is allowed to send the data type. If allowed, the DTE mechanism attaches a type label (attribute) to the data and sends it to its destination. Otherwise, an error indicator is returned to the requester.

When a process attempts to receive data, the DTE mechanism retrieves the type label attached to the data and determines whether or not the domain of the process is allowed to receive that data type. If the process is allowed, then the system returns the data to the process and, if requested, the data type. If the requester is not allowed, an error is returned to the requester.

Just as an application can send only one type of data per system call, it can receive only one type of data per call. If there is more than one type of data waiting to be retrieved from the system's input buffer, the system will return only the first type of data. Another system call is required to receive the next type of data.

We treat send and receive modes as being independent and uni-directional even though some forms of IPC inherently include "back flows" of control information from the receiver to the sender for flow control and reliable delivery. We intend DTE to be used primarily to improve integrity and reliability. Reliable communication, especially across "lossy" channels, almost always requires acknowledgment messages to ensure that data has been delivered. Consequently, our view is that these "covert" flows are essential to our larger purposes and should not be eliminated. Nor should send and receive operations be treated as equivalent simply because both can induce bi-directional information flow. From the standpoint of least privilege, the distinction between sending and receiving is essential.

3.3 DTE Network Objects

Next we consider the notion of message mediation in the context of two predominant styles of data communication in IP networks: datagrams and streams[10]. Datagram protocols preserve data boundaries used by a sender. The data presented at a single sending system call is treated by the system as a unit, referred to as a datagram. The receiver retrieves a single datagram in each receiving system call. In contrast, stream protocols do not preserve the data boundaries used by the sender. The communicating parties establish a connection to carry the stream of data between them. The sending and receiving processes can independently choose, at each system call, the number of bytes of data being relayed through the data stream. The objects in our model are typed messages, which are represented differently in the two styles of protocols.

For datagram protocols, typed messages are datagrams. Each datagram has an associated type. Although a datagram may become fragmented by lower layer protocols during transmission, the receiving system will reconstruct the original datagram before presenting it to the receiving process as a unit.

Since stream protocols do not preserve data boundaries, determining what constitutes a typed message is less obvious. Further complicating the issue, we want to allow processes in different domains to share streams, as is done in the common UNIX paradigm for communication illustrated in Figure 4. This figure shows the connection used for a remote login to another host. The remote command shell creates child processes, via the fork and exec system calls, to perform certain tasks. The command shell shares the connection with its children but temporarily refrains from using it until each child has terminated. In this example, let's assume that the user wishes to perform two

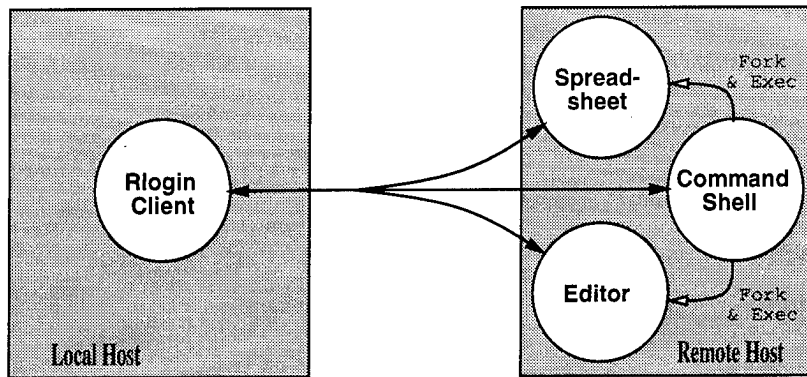


Figure 4: Connection sharing

unrelated tasks during a single remote login session: (1) using a spreadsheet to work on a project budget and (2) using an editor to update a personal mail message. According to the principle of least privilege, each of the processes should have access only to those objects essential to its function. Consequently, each should execute in a different domain and have access to different types of data. For backward compatibility, we would like to be able to use existing spreadsheet, editor, and command shell programs. Restricting a connection to carrying a single type of data during its lifetime would prohibit the spreadsheet and the editor in the example above from sharing the connection with the command shell.

Supporting this common UNIX paradigm is one reason our model allows stream protocols to carry more than one type of data. A second reason is to provide designers of network applications freedom to choose appropriate IPC architectures. In particular, we did not want to force designers to use separate connections for each type of data processed by each network application. A third reason is to provide symmetry between datagram and stream-oriented services by allowing a communications port of either kind to carry multiple types of information.

For stream protocols, our approach concatenates data from consecutive send operations as long as the type of the data is the same. We refer to this concatenated type-homogeneous portion of the stream as a *substream*. For stream protocols, the typed message is the substream. When the type of data being sent via the stream changes, a new substream is started.

A single receive operation for a stream protocol can return data only from within a single substream. Data from the next substream must be retrieved by a separate system call. Thus for stream protocols, when data are of the same type, the boundaries used by the sender are not preserved and the stream behaves exactly as it does for non-DTE systems. Whenever adjacent data from the sender are of different types, the type boundary is preserved but has no effect other than requiring the receiver to issue an additional receive call to cross the boundary.

3.4 Constrained Stream

A potential drawback of allowing a stream to carry multiple types of information is that a sender may inadvertently send data that is not receivable in the intended receiver's domain. Moreover, an incompatibility of this sort between the domains of the sender and receiver will not be detected

until the receiver makes its system call. To remedy this problem and allow earlier detection and handling of errors, we have introduced the notion of a constrained stream.

When a connection is established for a constrained stream, both parties must agree upon the type of data that will be transmitted during the lifetime of the connection. For a constrained stream, additional mediation occurs during connection establishment. The DTE mechanism verifies that each party has appropriate access to the type that it specifies for the connection and that the constraints specified by the two parties match. By making certain that each party has appropriate access to a jointly specified constraint type, the DTE mechanism verifies that communication is allowed between the two processes before it completes the connection. It then restricts each party to sending only the type of data that its peer has advertised and is allowed to receive. This allows processes to discover any data type conflicts before any data is sent. Nevertheless, type attributes are still carried with all data sent over a constrained connection.

Stream protocols typically allow processes at both ends of the connection to communicate bidirectionally, effectively creating two streams of data. To allow the type of data being sent by a process to be different from the type of data received by the process, a constrained stream in fact requires the selection of two data types, one for each direction. The input type of each end must match the output type of the other end.

4 A Network DTE Prototype

To investigate the practicality and usefulness of these abstractions, we have constructed a prototype DTE system based on OSF/1³ MK 4.0 UNIX. The DTE operating system runs as a server process atop the Mach microkernel [1, 8].

The OSF/1 UNIX server provides the socket services of Berkeley Software Distribution (BSD) UNIX [10] as the primary application interface for network and single-host IPC facilities. Although computer security researchers have considered the characteristics of UNIX IPC previously, their focus has been on issues associated with DoD MAC rather than DTE [16, 17]. Our DTE prototype extends the UNIX socket interface with additional DTE semantics and services. These extensions were carefully designed so that they apply as uniformly as possible to all supported protocols. For local communication, the DTE prototype currently supports UNIX Datagram and UNIX Stream protocols. For network communication, the prototype supports Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

4.1 Backward Compatibility – Default Creation Type

To accommodate new and existing application images, the prototype provides two styles of IPC system calls. New style calls allow new, DTE-aware programs to make full use of DTE features. New style send calls require the calling program to specify the type of data being sent. New style receive calls return the type of the data received.

Old style calls are binary compatible with existing programs that are not DTE-aware. Old style send system calls do not allow the caller to specify the type of data being sent. Instead, a system architect or DTE administrator may assign a default creation type to a domain. By default, the

³OSF and OSF/1 are trademarks of Open Software Foundation, Inc.

DTE mechanism associates this type with any data sent from that domain via an old style system call. An old style send request will be rejected if no default type has been established.⁴

4.2 Interoperability – Labels and Non-DTE Hosts

We chose the IP option space as the place to convey type labels passed between hosts. This allows the transport protocols (TCP, UDP) to share a common mechanism for carrying labels. Since they do not carry any DTE information, these protocols did not have to be changed. Each IP packet carries exactly one type of data; the packet's header carries the type label. When a single stream carries more than one type of data, the DTE mechanism forces a new IP packet to be created at the beginning of each substream. Since type labels are visible in the IP header, this approach gives future network layer devices such as routers and firewalls an opportunity to make routing or screening decisions based on DTE labels.

Communication with existing systems is enhanced by allowing the DTE administrator to limit communication with non-DTE hosts. The administrator assigns a domain to each non-DTE host or group of non-DTE hosts with which communication is permitted. The system treats each non-DTE host as though all its processes were in the domain assigned to the host. Each assigned domain must include a default creation type so that when an unlabeled packet arrives, a receiving DTE system can attach the appropriate type to it. Access to incoming data from a non-DTE host can then be mediated properly. When a process attempts to send data to a non-DTE system, the DTE mechanism performs its normal mediation on the sending process and then makes certain that the remote host's domain is allowed to receive the data. In this situation, the sending DTE system performs the mediation that would ordinarily be done by a receiving DTE system. This makes it possible to prevent certain types of data from being sent to specific non-DTE hosts. A DTE system, however, cannot prevent processes on non-DTE hosts from collaborating with one another to circumvent the intent of these restrictions; hence such facilities should be relied on only in an appropriate context.

To ensure interoperability with non-DTE systems, the IP headers of packets sent to non-DTE hosts contain no DTE attributes. Non-DTE hosts, of course, have no use for such information. More importantly, in spite of the fact that the IP protocol standard directs hosts to ignore IP header fields that are not applicable, we initially caused non-DTE systems on our LAN to crash by sending type attributes to them.

4.3 Experience With the Prototype

Our experience with the prototype, although limited, has been very positive. All of the network abstractions described above have been implemented without difficulty. Using experimental DTE-aware applications, we've exercised fine-grained control over both datagram and stream-based IPC and believe that the network DTE mechanisms we've developed can be equally applied to both with significant ease. These mechanisms are integrated with the same DTE policy specification language and mediation rule base as the mechanisms that control access to file system objects [1]. As a result, organizationwide constraints on interactions among processes and users can be enforced consistently whether these interactions are attempted via a single-host file system or network IPC.

⁴The prototype provides additional mechanisms for establishing defaults. A discussion of them is beyond the scope of this paper.

We have demonstrated backward compatibility with existing UNIX binaries by executing well-known network communication utilities (e.g., rlogin, telnet, inetd) in custom-tailored domains. We have also executed and confined sequences of file manipulation utilities (e.g., cat, more, grep) that communicate via UNIX pipes, which are built, in turn, on top of DTE-controlled socket services. We have demonstrated interoperability by keeping several DTE prototype systems used for demonstrations and software development connected to our operational corporate LAN for many months. Also attached to the LAN are over 200 other non-DTE systems including personal computers, UNIX workstations, and file servers. While the DTE systems exchange DTE-labeled IP packets among themselves, they also interoperate seamlessly with the non-DTE systems.

5 Summary

Access control mechanisms provided by UNIX and other operating systems often used in networks allow users and their programs to extend access rights freely to other, potentially unauthorized, users via network IPC and other services. For this reason, such systems cannot uniformly enforce organizationwide restrictions on access to sensitive information. While DoD MAC provides stronger protection, it is inflexible and not well-suited to enforcing site-specific security policies, particularly those aimed at integrity and role-based access control.

We have described techniques for providing strong, flexible, organizationwide control over IPC among networked computer systems. Our approach extends DTE, an enhanced version of type enforcement, to datagram and stream protocols in a consistent and uniform manner. A UNIX-based research prototype has been constructed and demonstrates that network IPC controls based on these techniques can be integrated easily with DTE controls for file system objects. Moreover, the prototype has proven to be backward compatible with existing UNIX network programs and interoperable with existing IP-based LANs.

Our plans for follow-on work include developing larger demonstration applications to further validate the usefulness of these techniques for supporting integrity and role-based policies. We are pursuing integrating our prototype with the Trusted Mach⁵ TCB [23] as an untrusted server to demonstrate that network DTE can be combined with high assurance DoD MAC. We also plan to incorporate network DTE features and cryptography into an Internet firewall so that the firewall can screen IP traffic according to DTE attributes and coordinate DTE policies with other firewall-protected enclaves of DTE hosts.

References

- [1] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, S. A. Haghghat, "Practical Domain and Type Enforcement for UNIX," in *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pp. 66-77, Oakland, CA, May 1995.
- [2] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, "A Domain and Type Enforcement UNIX Prototype," in *Proceedings of the 5th USENIX UNIX Security Symposium*, Salt Lake City, UT, June 1995.
- [3] D. E. Bell and L. Lapadula, "Secure Computer System: Unified Exposition and Multics Interpretation," Tech. Report ESD-TR-75-306, Electronics Systems Division, AFSC, Hanscom AFB, Bedford MA, 1976.

⁵Trusted Mach and TMach are registered trademarks of Trusted Information Systems, Inc.

- [4] K. J. Biba, "Integrity Considerations for Secure Computer Systems," Technical Report ESD-TR-76-372, USAF Electronic Systems Division, Bedford, MA, 1977.
- [5] W. E. Boebert and R. Y. Kain, "A Practical Alternative to Hierarchical Integrity Policies," in *Proceedings of the 8th National Computer Security Conference*, pp. 18-27, Gaithersburg, MD, Sept. 1985.
- [6] D. D. Clark and D. R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, p. 184, Oakland, CA, 1987.
- [7] T. Fine and S. E. Minear, "Assuring Distributed Trusted Mach," in *Proceedings of the 1993 IEEE Computer Society Symposium on Security and Privacy*, pp. 206-218, Oakland, CA, May 1993.
- [8] D. Golub, et al., "UNIX as an Application Program," in *Proceedings of the Summer 1990 USENIX Conference*, pp. 87-96, June 1990.
- [9] J. Ioannidis and M. Blaze, "The Architecture and Implementation of Network-Layer Security Under UNIX," in *Fourth Usenix Security Symposium Proceedings*, pp. 29-39, Santa Clara, CA, Oct. 1993.
- [10] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, *4.3 BSD UNIX Operating System*, Addison-Wesley, Reading, MA, 1989.
- [11] S. B. Lipner, "Non-Discretionary Controls for Commercial Applications," in *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, p. 2, Oakland, CA, 1982.
- [12] S. E. Minear, "Controlling Mach Operations for Use in Secure and Safety-Critical Systems," Technical report, Secure Computing Corporation, Roseville, MN, June 1994.
- [13] S. L. Murphy, D. F. Sterne, L. Badger, and D. L. Sherman, "Distributed Access Control for Dedicated Systems," TIS Technical Report 508, Trusted Information Systems, Inc., Glenwood, MD, Jan. 1995.
- [14] National Computer Security Center, "Department of Defense Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, Dec. 1985.
- [15] R. O'Brien and C. Rogers, "Developing Applications on LOCK," in *Proceedings of the 14th National Computer Security Conference*, pp. 147-156, Washington, DC, Oct. 1991.
- [16] T. J. Parenty, "The Incorporation of Multi-Level IPC into UNIX," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 94-99, Oakland, CA, 1989.
- [17] S. G. Romero, C. Schaufler, N. Bolyard, "BSD IPC Model and Policy," in *Proceedings of the 16th National Computer Security Conference*, pp. 97-106, Baltimore, MD, 1993.
- [18] J. Rushby, "Kernels for Safety?" T. Anderson, editor, *Safe and Secure Computing Systems*, chapter 13, pp. 210-220, Blackwell Scientific Publications, 1989.
- [19] J. Saltzer and M. Schroeder. "The Protection of Information in Computer Systems," in *Proceedings of the IEEE*, 63(9), March 1975.
- [20] O. S. Saydjari, J. M. Beckman, and J. R. Leaman, "LOCK Trek: Navigating Uncharted Space," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 167-175, Oakland, CA, 1989.
- [21] D. F. Sterne, "A TCB Subset for Integrity and Role-Based Access Control," in *Proceedings of the 15th National Computer Security Conference*, pp. 680-696, Baltimore, MD, 1992.
- [22] D. J. Thomsen, "Role-based Application Design and Enforcement," in *Proceedings of the Fourth IFIP Workshop on Database Security*, Halifax, England, Sept. 1990.
- [23] Trusted Mach, "Trusted Mach System Architecture," Technical Report TIS TMACH Edoc-0001-93B, Trusted Information Systems, Inc., Glenwood, MD, May 1993.

INTEGRATING COTS APPLICATIONS ON COMPARTMENTED MODE WORKSTATIONS

Susan A. Heath
The Boeing Company
7990 Boeing Court
Vienna, VA. 22182
703-821-6272
Heath@dockmaster.ncsc.mil

Abstract

Though UNIX has been around for 25 years, Commercial-Off-The-Shelf (COTS) applications for this operating system were not widespread until five to ten years ago. With the growth of the use of UNIX to support office environments, there has been a parallel growth in the number of COTS applications designed specifically for this operating system. During this same timeframe, implementations of trusted UNIX, specifically Compartmented Mode Workstations (CMWs) have grown substantially and are now available from a number of different vendors. The combination of these two technologies often results in significant challenges and sometimes surprising outcomes for systems integrators.

Keywords: Compartmented Mode Workstations, COTS applications, integration, untrusted applications, UNIX

Introduction

This paper discusses some of the issues which must be addressed when integrating COTS applications on a CMW implementation of the UNIX operating system. Because the operating systems discussed are CMWs, they provide all B1 features and some B3 features as documented in the Department of Defense Trusted Computer Security Evaluation Criteria [1] with additional features as documented in the CMW requirements documentation [2]. This includes, of course, a trusted X Window.

The COTS applications addressed include Office Automation suites (word processor, spreadsheet, graphics, and e-mail), desktop publishing packages, program management packages, and relational database products. None of these applications were developed for use with a Trusted Computing Base (TCB) and therefore, none provide any trusted features. With the exception of the database products, which provide an unevaluated form of Discretionary Access Control (DAC), none of the applications provide any form of security whatsoever.

One major issue in integrating applications onto a CMW operating system is to keep the applications COTS. This means not modifying the products specifically to support the trusted environment unless the vendor agrees to make the change a part of the COTS baseline product. This paper discusses how COTS applications were configured to execute on a CMW without modifications to the applications.

The issues with COTS integration discussed fall into several basic categories which are discussed below.

General Operation

In order to provide execution of an untrusted COTS application in a multilevel environment, the application must be executed separately at each classification level required. This does not require multiple copies of the source code, but does require the application to be executed separately at each classification level even if the user is already executing the application at another classification level.

To facilitate this, application daemons must be started at multiple classification levels at boot. This is accomplished by starting the application in one of the start up scripts with the appropriate environment variables in place once for each classification level required. For those applications which do not require daemons, the environment variables associated with the process must be configured in the security database so that they will be set correctly when a user executes the application.

The determination of the correct environment variables is very important to maintain the trustworthiness of the system. These include the classification level, owner, group, and privileges to be assigned to the application. Even though the system operates in a multilevel environment, it may not make sense to run certain applications at any other level than unclassified. An example of this is a calendaring program in an environment where calendars are not classified. In this case, the application must be constrained to the unclassified level only. In addition, the owner and group of a process are very important in determining which users can access the process. For example, all programs which must be executed by the system administrator should probably be in the sysadmin group to facilitate this. The privileges associated with a process are extremely important and are discussed in detail in the next section.

Privileges Required

In keeping with the least privilege principle, it would seem to be a particularly unsafe thing to provide an untrusted COTS application with privileges which allow it to circumvent the trusted features of the operating system. In general this is true and can be adhered to, but certain circumstances make this impossible. One example of this is allowing access to the default colormap provided by the X Window System.

In X, each window has an associated colormap that determines how the pixel values are translated into colors. Since the colormap is a resource of the X server, when it is started, it creates and installs a default colormap with two color cells. The rest of the cells can be allocated and used by any X application. An application can create its own colormap or use the default one created by the X server.

To use a color other than the ones in the default colormap, an application must request that the X server allocate a colormap cell with the desired red, green, and blue intensities in the colormap of choice (either the default one or the one created by the application). The X server will return an index that can be used as the pixel value corresponding to that color. Then when the application wants to use that color, it simply specifies that pixel value [3].

If the application creates its own colormap, no special privileges are required because the colormap can be created at the sensitivity level that the program is running at. However, in some CMW

implementations, the default colormap is an unclassified resource. Therefore, in order for an application running at a higher sensitivity level to be able to write to the map, a privilege which allows the program to "write down" is required. Unfortunately, this privilege allows an untrusted application to override some of the mandatory access control rules and must be very carefully controlled.

The risk of providing the untrusted application with such a powerful privilege is mitigated by ensuring that users cannot take advantage of this privilege in any way. This is discussed further in the section on shell escapes.

Another area where special privileges must be given to an application is when the application needs to access window resources for a window that it does not own. When an application is executed, it must determine the attributes of the root window (the background window) to determine things such as where to place itself since all windows opened are children of the root window. Unfortunately, the root window is owned by the X Server. This results in the application receiving an access error. Therefore, without special privileges, the application cannot acquire the necessary window information. In this case, the application must be given a privilege which allows it access to X resources associated with the root window.

Shell Escapes/Running the Application Directly from the Shell

In an attempt to provide user friendliness and compatible features with DOS counterparts, many products provide some form of shell escape from within the product. These range from providing a "Go To UNIX" option on a menu to allowing the user to enter "!command" from within a document or mail message. Shell escapes can be dangerous in a trusted environment because the potential exists for the user to be able to take advantage of the privileges given to the application.

Simply "banging" out to the shell does not result in a user inheriting the privileges of the application because the effective privileges at this point are the combination of the privileges given the user and the privileges given the shell itself (hopefully none!) [4]. However, there are circumstances where a user can gain access to privileges given the underlying applications. One such example is within e-mail.

Some applications allow user's to take advantage of a feature provided by sendmail by allowing them to place the string "!command" within an e-mail message. The sendmail process will interpret the line of a message beginning with "!" as a command and will execute it. This is very dangerous in the trusted environment because sendmail is a trusted process and runs with several privileges. A user can create a script to do many things which they would not normally have access to and then simply include the name of the script with a "!" in front of it in an e-mail message. The script then gets executed by sendmail with all of its privileges.

This problem is easily overcome by removing the capability within the applications e-mail feature and within sendmail to interpret the command string within the message.

In addition to shell escapes, allowing users to run applications directly from the command line should also be avoided if the application has been given any special privileges. This is because there is a great risk that the user can take advantage of the privileges given the application. For example, most

applications will allow a user to enter a command to execute the application followed by a document name. This results in the application coming up and bringing the user directly into the document. If the application has been given the privilege to override MAC (as we have seen is necessary sometimes), then the user could enter the command to execute the application followed by a document name which is at a classification level higher than that for which the user has access. Since the process runs with the privileges given the application unioned with the privileges associated with the user, the user will be brought into the document if Discretionary Access Controls do not stop the access. This is potentially a very serious situation and should be avoided at all costs.

Administration

The CMW system is designed to support a division of power. This means that the authority traditionally associated with the root account (or uid=0) has been divided among several roles so that no one person has complete control of the system. Simply having uid=0 does not provide the power it does in untrusted UNIX. The account must have the appropriate privilege necessary to perform the desired function. The roles are established through the use of command authorizations that can be given to any account.

However, to an untrusted COTS application, a command authorization is meaningless and most products check for uid=0 when product administration of any type is attempted. This means that product administration must still be accomplished by an administrator logged on as root. This completely overrides the intent of the division of power which was specifically designed into CMW systems and sets a precedent of using the root account which may not be acceptable.

To maintain the division of authority inherent in the CMW system but also protect product administration, the COTS product must have a means of allowing a uid or gid other than root access to administration of the product. This is most easily accomplished by allowing the product to be configurable such that other login ids (different from root) or group ids (such as sysadmin) can accomplish system administration. However, some COTS applications do not offer this capability.

Print Servers

Many COTS applications use the concept of a print server to allow users to print documents in a multi-user environment. The print server is a shared resource which performs the job of configuring the document for printing and sending it to the print subsystem. It can be configured to start the first time a user executes the application or to run as a daemon which gets started at boot. If it is started the first time a user executes the application, it remains active until there are no users using the application and eventually times out. While it is active, it is shared among all users currently executing the application. When all users exit the application, it times out and is started again the first time another user enters the application. Obviously if it is run as a daemon, it remains active until the system is shut down or it is killed. Either configuration can cause problems in a trusted environment.

If the print server is started by the first user who executes the application, it is owned by that user. In an untrusted environment, the executables associated with printing are owned by root and the set_uid

bit is on. When these are executed, the process assumes the effective uid of root which allows it to access the print server even though it is owned by another user.

In a trusted environment, without special privileges to allow the process to emulate the untrusted UNIX environment, the set uid bit has no affect because uid=0 does not have the same privilege it has in untrusted UNIX. This results in only the first user being able to access the print server. No other users can print anything until that user exits the application and the existing print server times out. Obviously, this is unacceptable.

If the print server is started at boot, all users can access it, because it is a daemon, but technically it is owned by root. This results in banner pages for all print jobs which have root as the user. While this is not as drastic a situation as the one described above, it is still a less than optimal solution.

What is needed is a way to start a print server for each user so that all users can access it and the print out banner pages reflect the correct user. This can be accomplished in several ways, the easiest of which is to start the application with a variable which tells it to start a print server for each user. Some applications have this feature as an alternative print scheme and some do not.

Directory Structure

The final area of COTS integration onto trusted UNIX to be discussed is that of directory structure. Many applications require a directory for each user which is associated with the application. All files associated with the application are stored here. This is convenient for all users because their documents are stored underneath their home directory. Most applications will also automatically create this directory in the user's home directory if it is not there when the user executes the application. This creates a problem for some trusted UNIX file system implementations.

CMWs use a file system implementation consisting of single level and multilevel directories. A single level directory is just that, a directory which has a single sensitivity level associated with it. In some implementations, only files of that sensitivity level can be stored there unless the user has a privilege to "write down" or "write up". The concept of a single level directory is illustrated in figure 1.

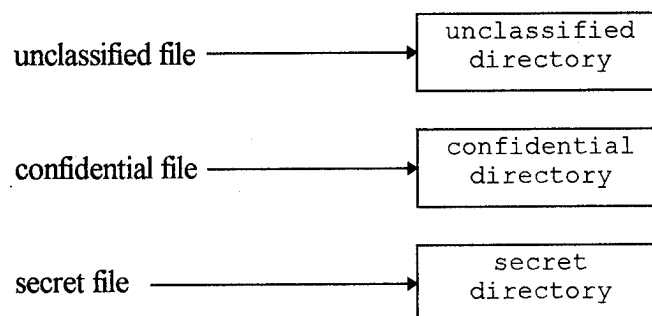


Figure 1 - Single Level Directories

A multilevel directory on the other hand can store files of any sensitivity level defined on the system. A multilevel directory is actually a parent directory with several hidden directories underneath, one for each valid sensitivity level as illustrated in figure 2. It appears to the user and untrusted applications as a single directory. When using an untrusted application in this environment, the directory structure must be carefully analyzed to determine which directories need to be multilevel and which can be single level.

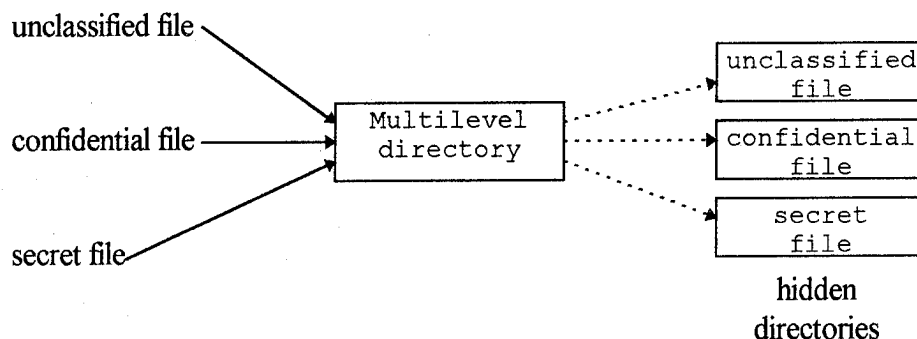


Figure 2 - Multilevel Directory Structure

In order for a user to be able to store files of varying sensitivities in the directory associated with the application, it needs to be a multilevel directory. However, an untrusted application has no knowledge of multilevel directories. When a user executes the application, if the needed directory is not there, the application will create it. Assuming the user executed the application at unclassified, this directory will be single level, unclassified which results in the user being unable to store classified files in the directory. If the user executes the application for the first time at a higher level, there are other problems which will be discussed later.

Getting around the problem of the application creating a single level unclassified directory is simple. Whenever a new user account is created, the associated multilevel directory is also created. Then when the application is executed, the directory is already there. Usually the operating system provides a method to tailor the construction of the user directory tree and configuration files such that they are automatically created whenever a new account is created.

However, there could still be a problem. When the application is executed, it may attempt to open the user's home directory not just for read, but also for write, in case it needs to create the applications directory. This is fine, when it is being executed at the unclassified level, but results in an access error at higher sensitivity levels if the user's home directory is single level unclassified. So even though the applications directory is already there and is multilevel, an access error will still be received when the application is executed at any level above unclassified because the application attempts to open an unclassified directory for write at a higher sensitivity level. If this is not a configurable item, which in many cases it is, some other solution must be found.

There are several ways around this problem. First, all users could have multilevel directories for their home directory. This is not a very good solution, however, because it results in the user actually having multiple home directories (in the hidden directories underneath the home directory).

Configuration scripts (e.g. .cshrc) might have to be replicated and maintained at each level. Since user's frequently tailor these files to their own liking, they would have to maintain multiple files. More basic than that though, is that users who may not be very familiar with UNIX will have to understand how the multilevel directory concept works in order for them to correctly use their home directory. This is a difficult concept for even seasoned UNIX users to understand, much less a novice user. It seems that this is not a good solution to the problem.

The other solution is change the \$HOME variable to the multilevel application directory which was created at account creation before executing the application. This could be done in a script and users would not have to be aware of it. The application would be tricked into thinking that the user's home directory was /usr/account/app_name instead of /usr/account and open this directory for read and write. Since it is a multilevel directory, no access error would be received. However, the application would then proceed to create its directory underneath this one because it thinks it is in the user's home directory. The applications directory would subsequently be created in each of the hidden directories underneath /usr/account/app_name as the user executed the application at each of the authorized sensitivity levels. This results in an extra layer in the user's directory tree (for /usr/account/app_name) and different directories (/usr/account/app_name/app_dir) at each authorized sensitivity level where one directory is really all that is needed.

Summary

There are many challenges when attempting to integrate untrusted COTS applications onto trusted UNIX. These include giving privileges where necessary, eliminating risks associated with shell escapes and running the application directly from the shell, administration, print server issues and directory structure complications. However, while there are many implementation details which are challenging when integrating untrusted COTS products onto trusted UNIX, there are few which cannot be overcome by giving privileges in a guarded way or configuring the application a little differently than it would be configured in an untrusted environment.

References

1. Department of Defense Trusted Computer Security Evaluation Criteria, DOD 5200.28-STD
2. Security Requirements for System High and Compartmented Mode Workstations, DDS-2600-5502-87, John Woodward, November 1987
3. X Window System Programming, Nabajyoti Barkakati, SAMS, 1991
4. CMW+/386 Trusted Facility Manual, SecureWare, Inc. Part number 010-00088-00, Revision E, April 1992

PROJECT WINMILL:

Using a COTS Solution to Connect LANs of Different Compartments

Mr. Al Nessel and Mr. Curt Sawyer
Defense Intelligence Agency
Advanced Technology Laboratory (SY-1C)
Building 6000, Bolling AFB
Washington, D.C. 20340

Executive Summary

The Trusted Windowed Information Networked Multilevel Interconnected Labeled LAN (WINMILL) project provides a much desired capability, namely the connection of two Local Area Networks (LANs) operating at different compartmented levels. For convention, the two networks joined are a TS:A/B LAN and a TS:A/B/C LAN. This new connectivity is another step towards the goal of one workstation accessing all information sources.

The WINMILL system comprises one Network Information Service (NIS) server, one Trusted Label Router (TLR), and some number of client workstations, all installed on the "high" side (i.e., the TS:A/B/C LAN) in multiple sensitive compartmented information facilities (SCIF). Each component runs Sun Microsystem's Trusted Solaris 1.1. Trusted Solaris is Sun's implementation of the Compartmented Mode Workstation (CMW) Requirements. It is currently under National Computer Security Center (NCSC) evaluation for accreditation as a B1+ CMW.

WINMILL takes advantage of the security features inherent in Trusted Solaris. Trusted Solaris allows users to operate at different session levels and appropriate compartments depending on their clearances. MAXSIX is Sun's trusted network implementation of DoDIIS Network Security for Information eXchange (DNSIX), and it secures and limits network traffic between two classified but independent networks.

The WINMILL architecture supports TS:A/B network traffic between the A/B/C LAN and the A/B-LAN. Initially, this is tn3270 application traffic and SMTP e-mail traffic. TS:A/B/C data is processed on the A/B/C LAN but is unable to reach the A/B LAN.

Disclaimer

Project WINMILL is a specific application of CMW technology to solve a specific problem. Limitations and restrictions described here are due to the project restraints, not the CMW technology.

Purpose of the WINMILL System

Project WINMILL grew out of a simple requirement. Personnel on a small, application-specific Top Secret Compartmented LAN (TS:A/B/C LAN) wanted to run an additional application from the larger Top Secret Compartmented LAN (TS:A/B LAN), and wanted to exchange e-mail with the users on that LAN. Since the two LANs are physically separate -- a security requirement since one processes an additional compartment called "C" -- this requirement might seem impossible to fulfill.

With the use of Compartmented Mode Workstation technology, however, the problem becomes quite simple. Users on the ABC side with this requirement could receive CMWs, which would inherently separate the ABC and AB data. With the addition of another CMW used as an IP security label-based router, the basic problem is solved. The router, containing two ethernet cards, only permits A/B traffic to flow out of the A/B/C LAN. All traffic can flow into the A/B/C LAN, however, since its compartments dominate the A/B LAN. The final solution is a little more complicated, but that is the basic idea.

System Description

a. System Name and Location

As further described in the System Architecture section that follows, the WINMILL system has components physically connected on one large fiber optic ring. The locations of the workstations are A/B/C SCIFed areas, while the TLR is in a separately controlled machine room. Therefore, interconnection of the A/B and A/B/C LANs physically takes place in the separately controlled machine room.

b. System Architecture

The WINMILL system architecture is depicted in Figure 1. This architecture allows access to the A/B LAN from the A/B/C LAN. Procedures are discussed in greater detail in this section. The architecture of WINMILL consists of hardware, software, and communications elements, which are further described in the following paragraphs.

(1) Hardware: Figure 1 depicts the hardware and communications architecture of the WINMILL system. The NIS server is a Sun SPARCstation 2 64 MB workstation with two 424 MB internal disks and has a 1.2 GB Database server option with a 2.2 GB tape backup unit. The client workstations are also SPARCstation 2s but with only one 424 MB disk and 32 MB RAM. All are attached to the A/B/C LAN via ethernet connections. The TLR is a SPARCstation 2 with 32 MB

RAM, 424 MB disk, and two ethernet adapters. The two adapters are configured to allow the TLR to act as a router between different IP subnets. This is the critical component in securing data transfer between the two LANs. It is important to note that "postmaster" and "mainframe" represent A/B LAN systems that need to be addressable to the TLR and Trusted Solaris machines. Their description is only noted due to the significant role they play in providing e-mail and application access.

(2) Software: The operating system on all WINMILL components is Trusted Solaris 1.1. There is no additional system software required. Planned for the near future is the integration of the System Acquisition Support Services (SASS) software into WINMILL. SASS software will provide the same office automation suite to the Trusted Solaris users that the DIA UNIX users now have. SASS software integration has no impact on the WINMILL function, but will add office automation services that will be useful to the analyst.

(3) Communications: The communications architecture for the WINMILL system is based on MAXSIX between Trusted Solaris systems, and normal TCP/IP between Trusted Solaris and non-Trusted Solaris systems.

For both the A/B and A/B/C LANs, the physical network is a fiber optic ring of repeaters. An ethernet cable runs from the repeater to the workstation. Connectivity from the A/B/C LAN to the A/B LAN will be through the approved TLR only. The TLR router will control authorized access to the A/B/C LAN and its resources. Communication protocols used in WINMILL are MAXSIX and standard DoD protocol sets (NFS, FTP, SMTP, TCP/IP). The A/B/C LAN is an IP subnet isolated from the A/B LAN via the TLR.

The A/B/C LAN has other components besides WINMILL. It is a fully functional operational LAN with several application systems present. These systems are currently available to the other A/B/C LAN users and will be available to the WINMILL system. These systems will remain at the TS:A/B/C level.

c. Systems Operations

WINMILL was designed to provide a fully functional LAN environment and not inhibit the user from doing normal activities as performed on the A/B LAN. The WINMILL environment consists of an NIS/office automation server which exports the user's home directory, mailspool, applications, and auditing directory. Permissions and accesses are strictly controlled by the Information Systems Security Officer (ISSO) and Administrator (ADMIN) for the A/B/C LAN.

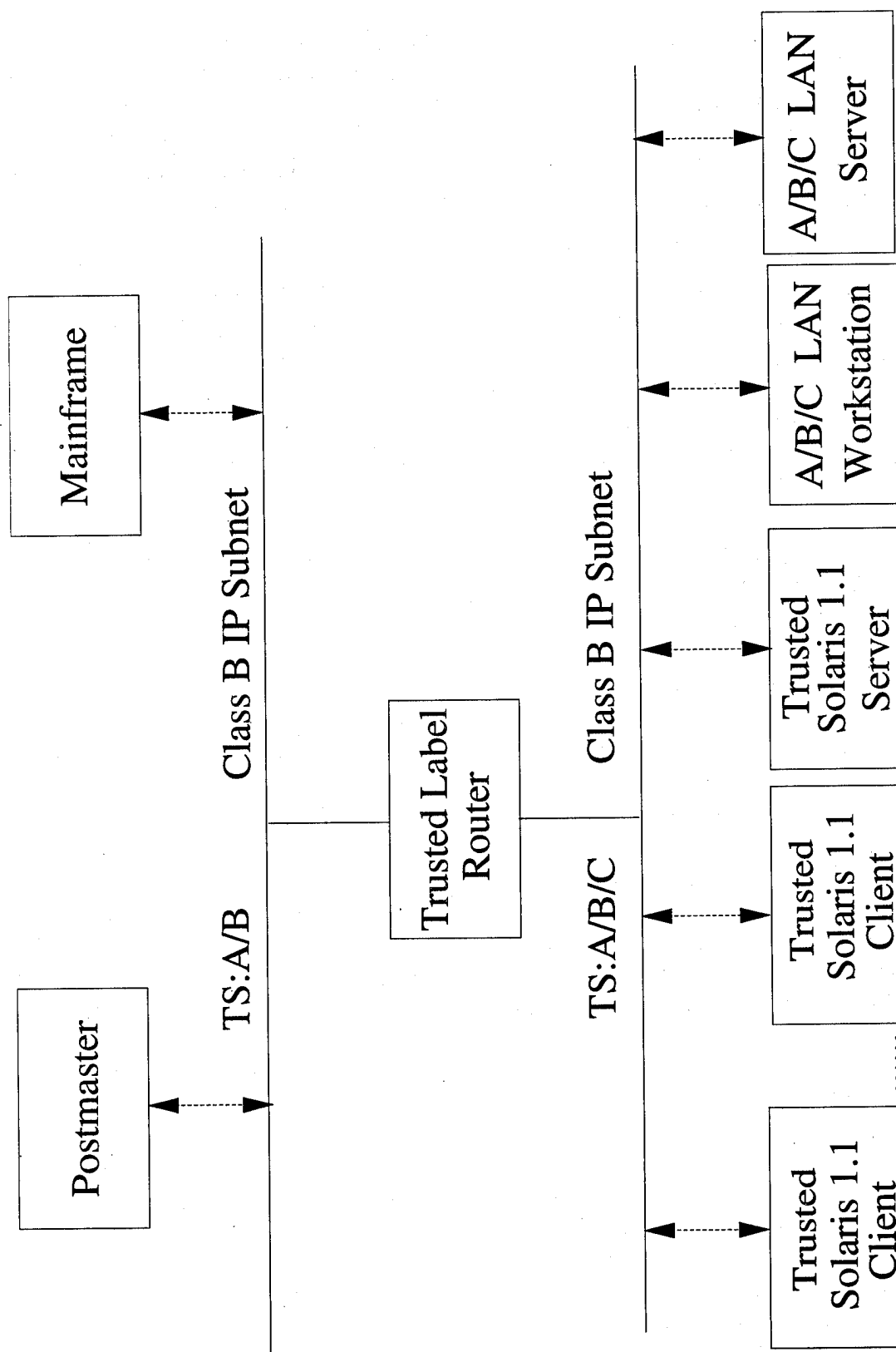


Figure 1.

The following scenario depicts the WINMILL setup and operations:

(1) Login, Authentication, and Session Level: A user will login to the Trusted Solaris NIS domain with a unique UID and password. This domain consists of the server or any of its clients. It does not apply to the TLR. (The TLR will not be part of the server/workstations' NIS domain; it will have separate login accounts.) Once the user has logged into the domain, a session window prompts him or her for the highest session level he or she wants to be able to work at for which he or she is cleared. Normally, the user will select TS:A/B/C.

(2) User Environment: The user interfaces with the system through the desktop manager. Icons represent all user functions available through the desktop manager. The user simply clicks on the icon to invoke the application. What is important to note here is the sensitivity label at which the application is invoked. Under WINMILL, Trusted Solaris allows the user to invoke office automation applications at any level from UNCLASSIFIED up to TS:A/B/C. Any resulting document or file created will also be at that level. These documents will be stored in a multilevel directory (MLD). This directory mechanism allows the user to transparently drag-and-drop file icons onto the MLD icon and store the file at the appropriate level. The user is reassured of the level he or she is operating at by the sensitivity label banner associated with each window. The user can change the sensitivity label through the Trusted Path Menu Selector. This mechanism provides flexibility for the user to invoke applications or shell windows at the various classifications at which he or she is allowed to operate.

(3) A/B/C LAN to A/B LAN Network Access: Trusted Solaris enforces a policy that each workstation node must have in its /etc/host table and its /etc/security/TNETRHDB file entries for all the hosts it can communicate with. (There is no trusted implementation of Domain Name Service (DNS)). For each entry in /etc/host, there exists a corresponding entry in the /etc/security/TNETRHDB file, where the host is identified by a "MAXSIX" or "unlabeled" entry. A MAXSIX entry identifies the machine as another Trusted Solaris workstation that is capable of receiving MAXSIX-labeled IP packets. An unlabeled entry indicates that only normal IP packets can be sent to this machine. In addition to these two entries, the machine has an entry for its classification level. For example, the machines "mainframe" and "postmaster" are two essential nodes WINMILL allows connectivity to. Both have entries that show they are unlabeled and both are at the TS:A/B level. This prevents any user from accessing these systems from a sensitivity-labeled window other than TS:A/B.

The TLR's tables are identical to the other WINMILL machines. The TLR, as stated earlier, has two ethernet interfaces and routes unlabeled IP packets from the

A/B/C LAN to the A/B LAN. Both LANs are class B IP subnets. The routing is static and does not require manual intervention.

(4) A/B/C LAN Network: WINMILL addresses the communications between the A/B/C LAN and A/B LAN from a Trusted Solaris workstation. Since other A/B/C LAN system-high workstations (SHW) exist on the A/B/C LAN, it is worth noting their interoperability with WINMILL. They will continue to access A/B/C-specific applications running on system-high servers. They are not configured for communicating with the A/B LAN. In fact, they are not able to communicate with the WINMILL machines. The WINMILL machines, however, are able to access the A/B/C-specific application servers, and can thus run the applications. (This configuration is specific to this project, and not a limitation of the CMW technology.)

(5) Summary of Functions: In summary, an analyst on a Trusted Solaris workstation can: (See Figure 2.)

- Address another Trusted Solaris machine from the UNCLASSIFIED to the TS:A/B/C level.
- Send and receive e-mail and enclosures at TS:A/B to any user on the A/B LAN
- Access the mainframe system at TS:A/B
- Access A/B/C-specific applications at TS:A/B/C

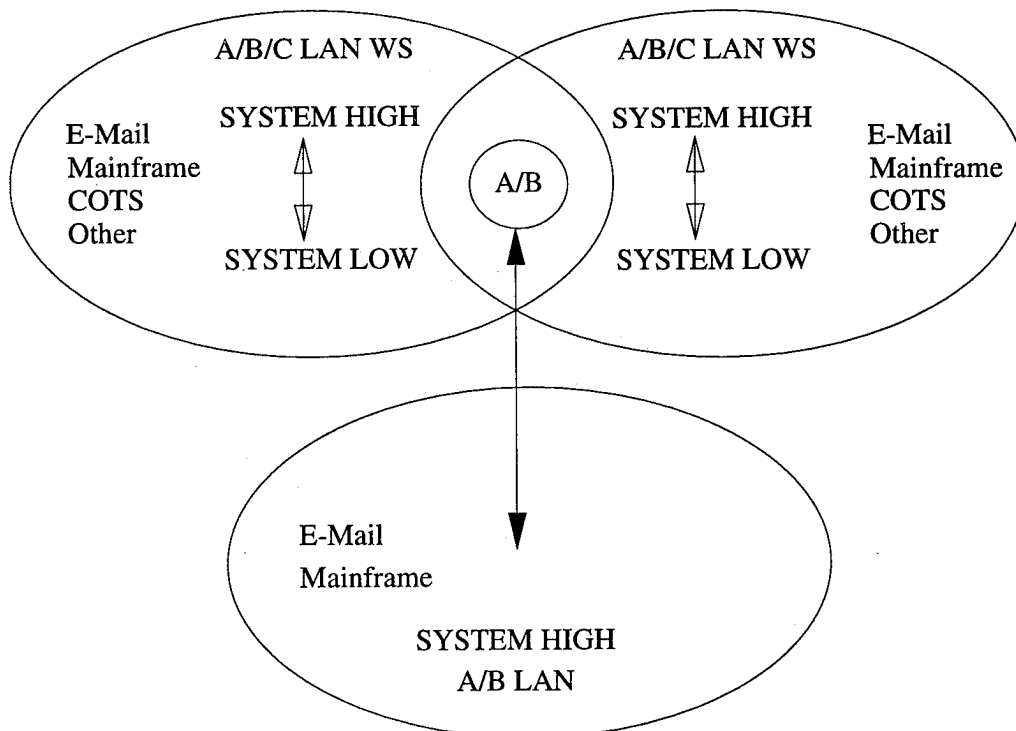


Figure 2.

d. Upgrading and Downgrading Information

Trusted Solaris allows upgrading and downgrading of information. Upgrading of information is allowed by the user. When the user upgrades information, a message window appears notifying the user of the action. The user confirms the action and proceeds with the upgrade. Currently only the ISSO can downgrade information. The ISSO can process requests from the user and accept or deny them, but the entire operation is controlled by the ISSO. This process is also audited by WINMILL. (In the future, the ISSO may grant highly trusted users the privilege to downgrade information, but this situation is not part of the original WINMILL project.)

Current Status of WINMILL

This project went from concept to operational test in about three months. The operational testing of WINMILL was completed in February 1995. Deemed a success by the customer and the security office, the project was put on hold pending completion of the security documentation. In the Laboratory the immediate future for WINMILL is the integration of as many of the SASS software products as possible. SASS integration should be completed before the security documentation for accreditation is completed.

Future Direction

This project filled a specific requirement while also finding a niche for CMW technology. In addition to joining LANs of different compartments, LANs of different levels could be joined, e.g., a SECRET to TOP SECRET, or a TOP SECRET to a TOP SECRET SCI. The technology allows us to build these labeled routers, but it is ultimately up to the security office as to when, or if, they are ever implemented.

ACRONYM LIST

ADMIN	System Administrator
CMW	Compartmented Mode Workstation
COTS	Commercial Off-The-Shelf
DNS	Domain Name Service
DNSIX	DoDIIS Network Security for Information eXchange
FTP	File Transfer Protocol
ISSO	Information Systems Security Officer
LAN	Local Area Network
MAXSIX	Sun's implementation of DNSIX
NCSC	National Computer Security Center
NFS	Network File System
NIS	Network Information Service
SASS	System Acquisition Support Services
SCIF	Sensitive Compartmented Information Facility
SHW	System High Workstation
SMTP	Simple Mail Transport Protocol
TLR	Trusted Label Router
TNETRHDB	Trusted Network Remote Host Database
WINMILL	Windowed Information Networked Multilevel Interconnected Labeled LAN

On Guards . . . En Garde

Lawrence M. Sudduth

Secure Computing and Communications, Inc.
P. O. Box 551
Great Falls, Virginia 22066-0551

Abstract: The information security issues associated with implementing a guard processor, e.g., the interface between domains processing at disparate system-high sensitivity levels, are discussed. A distinction is made between guard functionality which counters low-side penetration threats, and that which attempts to prevent information spillage, especially in a free-form messaging environment. The resistance of a guard to penetration attempts is directly proportional to the Trusted Computing Base's assurance level. The potential for inadvertent high information spillage can be reduced with the implementation of an automated filtering application on the guard, in which the guard's mandatory security mechanisms ensure that objects are filtered. The quality of the automated filtering mechanisms cannot approach that offered by manual review or processing sensitivity labels provided by B-division (and better) trusted multilevel secure systems, when these have been employed on the high side for data object creation. In the absence of trusted object viewers, only text can be reviewed, manually or otherwise. Establishing accountability for intentional information disclosure from high-side originators to low-side recipients could prove to be difficult, if not impossible.

Guard (gärd)

—v. tr. 1. To protect from harm or danger, esp. **by careful watching; keep secure**. 2. To watch over **to prevent . . . indiscretion**. 3. To . . . **supervise entries and exits**.

—n. 1. a. An individual or a group that **stands watch or acts as a sentinel**. 8. A **device or apparatus that prevents . . . loss**

Background

The increasing commercial availability of unattended E-mail guards, and other guards which purport to allow the secure exchange of unformatted data objects without operator intervention, has made life interesting for computer security practitioners. The marketers of these guards generally tout them as the "answer" to the operational requirement to exchange low information between a high (system high) environment and a low environment. Since the guard application is often implemented on a trusted, multilevel secure host, many believe that the guard

represents a solution to the security issues related to the interconnection of the domains. In point of fact, it does not. Rather, the guard offers a platform upon which a mandatory security policy can be enforced, and a capability to filter data objects, to some extent or other. Developing the security policy for the guard to enforce, which addresses the information security vulnerabilities associated with system-high operations, remains problematic.

Commercial guards target the widespread operational requirement to exchange low information, especially E-mail, between high-

side and low-side users. They generally offer extensive audit capabilities, some form of activity journalization, and a guard application with a palette of automated filters. At the very least, the guard filters have a "dirty word" identification capability. The presence of a specific text string (the dirty word) within an object causes that object to be enqueued for human review or to be scratched. Some guards also allow filtering of low-side input, to reject input which does not meet preestablished constraints.

Trusted Guards

Guards are employed at the interface between computing environments or domains that process at disparate sensitivity levels. Guard functionality can be implemented procedurally, or with hardware, firmware, or software. Media transfer (the so-called sneaker net) implements guard functionality; it often represents the only approved channel between domains. In order to counter the vulnerabilities associated with multilevel operations without a mandatory security policy enforcement mechanism, the media involved in the exchange often undergoes stringent review procedures. Administrative and procedural requirements associated with the media-review attempt to ensure that a mandatory security policy is enforced. Users are prone to violating these requirements, since they do not comprehend the technical vulnerabilities which are being countered. Media transfer can harbor all of the security vulnerabilities associated with a network connection, albeit at much lower bandwidth.

One of the primary requirements of guards is that they be reliable enough to enforce the applicable security policy with an acceptable level of residual risk. The risk is often multifaceted; an executable object which is transferred from the low to the high side can have far greater security consequences than the information content of a low-side data object being transferred to the high side. The information content of a (putative low) data

object transferred from the high to the low side also has security significance. The requirement for reliability of guard operation is often realized through the use of trusted multilevel secure systems (i.e., division B trusted computing bases), since these systems provide mechanisms (and varying levels of assurance) that a security policy will *always* be enforced.

A well-defined information control objective is a hallmark of most of the publicized guards which have been developed within the Department of Defense and the Intelligence Community in the recent past [Def94]. If designed for unattended operation, the guards use rigorous, rule-based filtering on highly formatted, input objects to validate or produce objects which are eligible for low-side processing. Upon successful completion of all required filter events, as well as any transliteration processing, the object's sensitivity label is downgraded (i.e., the mandatory access control label is regraded from high to low) and the object is eligible for transmission to the low domain. If the information content of the high-side, input objects cannot be modeled, an operator is generally in the loop, i.e., manual review is employed.

Some of the guards which have been implemented within the classified processing community run on separate physical machines with high assurance levels.

The Worldwide Military Command and Control System (WWMCCS) uses a formally rated (B3) multilevel system (XTS-200 running on Honeywell minicomputers) as a guard processor between the Top Secret system-high WWMCCS environment and certain Secret system-high environments. The guard application allows automated sanitization and downgrading of highly formatted data files originating from a Top Secret database, as well as other file transfers, many with manual review components. Since (doctrinally) a B1

system does not have the requisite level of assurance to protect classified information from a determined, technically qualified penetrator with a Secret clearance, the B3 guard processor controls the interface between the environments.

Other lower assurance guards have been developed/implemented. Where fielded, the accreditors have generally had to assume a higher level of residual risk, although ancillary protection mechanisms, such as shielding the low side interface from all but predicted communications with a trustable router, are often employed.

The Radiant Mercury guard uses a formally rated (B1) multilevel system (HP BLS on Hewlett Packard workstations) to host a guard application which transliterates and reformats highly formatted input SCI telegrams to output telegrams with lesser handling requirements.

The Operations/ Intelligence Workstation (OIW) uses a Sun CMW (SunOS, CMW Version 1.0, which is nearing the completion of its NCSC B1 evaluation) to implement a "human in the loop" guard application. The mandatory access control (MAC) policy of the CMW is used to ensure that putatively collateral information from the SCI high side, is only transferred to the low side after being reviewed and downgraded by a cognizant human reviewer.

Electronic Mail Guards

Several E-mail guards are available today, or will be available in short order, which enable E-mail exchange between environments operating at disparate sensitivity levels. These guards can generally be divided into two genera, which are unique in their concept of operation. The easiest way to distinguish the two from each other is by their marketing claims.

The first genus, *guardus consentrazioni Infosecum*, is marketed as a tool to assist in countering the perceived information security (INFOSEC) threats. Potential ancillary countermeasures are identified, and residual risks are described. The second, *g. c. Marketini*, boldly heralds the operational benefits and cost savings — statements which are in and of themselves, indisputable. The former class of guards are challenging to implement and operate; this is less the case with the latter class of guards. The sales pitch for the *g. c. Marketini*, guards generally includes some wording to the effect, "that the sponsors must assume some level of risk." This is certainly understated.

Risks Associated With Automated Guards

In an ideal world, security practitioners would only have to confront implementing guards between trusted multilevel secure domains. Data objects would have MAC labels which accurately reflect information content. These labels would be exported with the object so that they could be used as part of guard processing. Since the trustability of MAC labels is associated with the assurance level and accessibility of the infrastructure and other issues, this is not a silver bullet to INFOSEC concerns. Nonetheless, such an MLS enclave can be connected to another enclave through a guard. If a uniform threat and risk environment existed in both enclaves, the guard could be a trustable Internet Protocol (IP) router. If these environmental considerations do not hold true, the guard must be implemented on an MLS platform with the requisite assurance level to counter the penetration threats from the low side.

In a generally benign environment, like an enclave in which Secret- and Top Secret-cleared users originate Confidential through Unclassified information using B1 systems, the quality of the labels generally does not introduce significant residual risk. The guard could import the label with the object, process

the information contained in the object as required (to counter application-specific vulnerabilities), regrade the object (e.g., from Low-Unfiltered to Low-Filtered, since the guard's MAC policy enforces the security policy) and pass it low. The level of threat of the destination environment should play a large role in determining the trustability, or assurance level, of the guard.

The operational requirement most often faced today is to enable the interconnection of a high and low PC LAN enclaves for low E-mail exchange with an acceptable level of residual risk. When interconnecting domains which operate at different system-high levels, five broad categories of risk exist: high information can be transmitted low (spillage), low executables can be transmitted high (Trojan horse), the guard can be penetrated (enabling the previous two), and high information can be transmitted from the low side (disinformation or security violation). The fifth general risk is that user accountability for high information spillage can be difficult to establish.

High Information Transmitted Low

The automated guard must capture messages which contain high information, preventing them from being transmitted to the low side. To accomplish this task, data objects are filtered on the guard during transit. Ideally, guards are based on trusted MLS systems. They have separate ports connected to the high and low domain. Both of these ports (or devices) are labeled to reflect the domain to which the port is attached. If a min/max capability exists as far as device labeling is concerned, the minimum should equal the maximum, since the domains are system-high and -low respectively.

The port attached to the high side assigns a high label to incoming data objects. The TCB's mandatory security policy ensures that this object can never be written to the low port until its label is downgraded. In a perfectly secure world, a cognizant individual would

review the message using an application which is trusted to display the entire object contents [McH85]. If the object met the review criteria this individual would then exercise her downgrade authority, and change the MAC label on the object to low. The low object would then be eligible for low processing, the end result of which is that the message would be transmitted to the low domain. The volume of traffic which transits the guard, resource constraints, and other issues can all contribute to the decision to take the human (the only entity qualified to cognitantly review free-form text) out of the review cycle.

While each guard is unique, most operate as follows. The message is imported, and labeled high. Filters (running high) are used to review the message. Upon completion of all required filtering processing, the MAC label is changed to low. The message is then sent to the low domain.

While filtering is challenging for text files, it is extremely difficult for other file types, especially application data files which originate on DOS/ MS-Windows™ PCs operated in the Secret system-high mode, for example. Several issues are provided for illustration; this is not all-inclusive.

Even text-based, Windows-application data files all have inordinately long prologue sections, often one to four kilobytes in size. The prologue is full of application-specific information, which displays as control characters when viewed with a disk sector editor. Additional control information is present throughout the file. Without a mapping of the control information zones and the potential legitimate values, this data cannot be reviewed to determine if it contains encrypted representations of high information.

Windows applications generally support a file format which allows edits to be appended to the existing file. In MS Word®, for example, "Fast Save" an option which is

enabled by default, produces files which are up to 40% larger in size than files which are written completely anew when they are saved. Text and other data structures which have been deleted from the original document, is still present in the object, it is just not displayed in the application. This is especially noticeable with files which have been repeatedly edited and saved. The old version of the text is generally invisible within the application, but is visible if reviewed with a disk-sector editor, or even the MS-Windows NOTEPAD applet.

Windows applications which support Object Linking and Embedding (OLE) can produce data files which are actually compound data structures. OLE- and application-specific control information which is present in these files is almost incomprehensible to a reviewer. This data cannot be reviewed to determine if it contains encrypted representations of high information.

Information contained within non-textual files must be reviewed or filtered. The availability of a trusted viewer for text-based application data files is problematic. The problems can be insurmountable for other files including graphics, images, and spreadsheets. Even viewing the sector contents with a disk-sector editing utility will not shed light on the information contained in other than text files. Recent work [Kur92] indicates that files can contain contents which are invisible even to what is believed to be a trusted viewing application.

As hard disk size increase, the minimum physical space allocated to each file (the cluster size) also increases. As cluster size increases, the likelihood increases that the slack space between the end-of-file and the end-of-cluster contains remanent information. The slack space can be as long as eight kilobytes or more. If desktop machines are C2 or higher TCBs, and object reuse mechanisms are activated, this issue can be ameliorated.

Some of the representative vulnerabilities described above can be countered at the desktop machine, such as by zeroing out file slack space. The other vulnerabilities present potentially significant challenges. As unattended communications links are implemented between high and low enclaves, the predictability of the traffic (i.e., the content of the link) and the bandwidth of the medium contribute to the creation of an exploitable storage channel. The requirement to ensure that encrypted representations of high information never leave the high environment precludes the exchange of all but textual information, which by definition, can be reviewed by humans.

Low Executables Transmitted High

Mandatory access controls are required for multilevel processing to preclude Trojan horse attacks, among other reasons. When a high and low environment are interconnected, via magnetic media, telecommunications link, or network connection, a mandatory policy must be enforced to preclude the migration of low executables to the high side. This is the only way to prevent Trojan-horse attacks when interconnecting system-high environments. The executable code could be an object file, an application data file with an embedded macro, a script file, etc. As application macro languages become more sophisticated, the potential for covert attacks increases.

Guard Penetration

If the guard is to enforce a security policy, it should be robust enough to withstand subversion attempts. Is a CMW equal to this task? Probably not. This is not to say that CMWs are not secure. CMWs were designed to facilitate the analysis of information managed within separate compartments, by TS-SCI cleared intelligence analysts, such that the analyses produced were marked appropriately for control and release. They

were not designed to withstand determined penetration attacks [Ber90].

With the exception of one guard, the Trusted Multilevel E-mail Guard® (TMEG®) from Trusted Information Systems, Inc., the commercially available message guards are implemented on B1 Trusted Computing Base (TCBs) or Compartmented Mode Workstations (CMWs, B1+), at varying stages of National Computer Security Center (NCSC) evaluation. The robustness of these guards, i.e., their resistance to penetration, is a function of the underlying TCB [Dep85]. In light of this, the guard should be protected from low-side penetration attempts. Often this involves the installation of a single purpose, "trustable" router, to condition the link to the low side network.

TMEG, implemented on the B2 TCB Trusted Xenix, is the only E-mail guard which could potentially not require additional architectural measures to protect it from low-side penetration attempts. A successor version of this product will be hosted on the B3 TCB, T-Mach. This successor product meets the doctrinal outlook for acceptable risk [Com85] in a closed environment processing Secret to Unclassified information with an American user population ranging from uncleared to Secret-cleared. This product could be installed without architectural safeguards, external to the guard itself.

High Information Originating Low

Two different concerns relate to the ability of the guard to transmit high information from the low side to the high side. The first concern is that cleared users with no access to classified systems will avail themselves of the communications channel to transmit high messages to correspondents. This has been an issue with telephony for some time. The voice conversation is transient. Once it is completed, it no longer exists, unless it was intercepted and recorded. The issue is more complicated with messaging, since media

records of the message (which can legitimately exist on all of the intermediate systems between the source and the guard) increase the potential for compromise.

The second concern relates to disinformation attacks. The sanctity of the address space in E-mail systems is notoriously poor. Impersonation has caused many amusing anecdotes about why E-mail should not be used for formal traffic. Nonetheless, E-mail is being employed by users to assign and respond to taskings, report information, etc. While the decline in the amount of formal communications (telegrams) when an E-mail system is installed can be attributed to the off-loading of informal traffic, this is not always the case. This sets the stage for disinformation attacks, that the guard should attempt to counter.

Establishing User Accountability

One of the cornerstones of our (National defense information) security policy is to establish user accountability for their actions. If accountability was not required, identification and authentication to the individual level, audit, and other security mechanisms would not be required. As system-high environments exchange information with low environments, how can the user who inadvertently, or even deliberately, transmits high information to low users be held accountable for their actions?

I maintain they cannot. Every publicized, successful prosecution of an individual for espionage relied on incontrovertible evidence that the information was exchanged by the accused to the recipient, generally an agent handler employed by a hostile intelligence service. Especially after viewing portions of a celebrated murder trial, I fear this proof is not possible in the operational scenario described herein, in which electronic messaging is used to transmit the information. Even if sufficient accountability information is collected, and securely stored, on the both the high-side and

the low-side message transport systems, the high-side user's lawyer can claim an inadvertent act occasioned by system-high operations resulted in the information transfer. The low-side recipient's lawyer can claim the (low) environment security policy did not prohibit the receipt of information by uncleared individuals.

Since this issue is not the focal point of this note, it will not be dwelt upon. New statutes will attempt to resolve this conundrum, but security practitioners could still be grudgingly forced to be hostile witnesses for the defense, instead of prosecution witnesses.

Commercial Guards

SecureWare, Inc. markets the Secret-Unclassified Network Guard (SUNG). The MISSI program office markets a solution set for Fortezza encrypted E-mail from Secret to Sensitive but Unclassified domains. The TMEG product was previously mentioned. All of these have one trait in common — they attempt to meet a valid operational requirement securely. Two products are discussed below.

IM SUSPECT — *Information Migration between Secret and Unclassified Systems, Primarily Enforced by Conventional Techniques*

While the acronym is perhaps fitting, this E-mail guard doesn't exist. It exists in spirit in a number of installations where a security policy is enforced which counters legitimate technical threats. If the product did exist, it would only allow the exchange of reviewed textual E-mail messages, like Simple Mail Transport Protocol (SMTP). Since the reviewer is charged with ensuring that only proper information exchange occurs, each data object is reviewed with a trusted viewer in its entirety prior to transmission.

Since E-mail is generally free-form text, it is almost impossible to determine whether it is classified unless the information is read by a

cognizant individual. The necessity to only support textual E-mail from high to low is a direct result of the inability of most users to differentiate the control information in a standard word processing document, from a storage channel (e.g., encrypted representations of high information). The potential for detecting such signaling in a non-textual file, like a spreadsheet or graphic file is even greater, so these could not be transmitted from high to low.

The problem associated with low objects entering the high side is mainly trying to identify penetration attempts. It is difficult to distinguish between the embedded macro within a Lotus® 1,2,3 spreadsheet, (which can invoke almost any functionality) and the legitimate mathematical operations of an economics forecasting model. Script files are text-based; if the script is invoked on the high side, an insecurity may result. These are relatively easy for a human to detect.

While E-mail exchange is possible in this environment, it is resource intensive. E-mail which does not meet the review criteria would be rejected.

SMUG ENUF — *Secret (Multilevel) Unclassified Guard, for E-mail, News, and Unformatted Files*

This "product" doesn't exist either, except in the headlines and text of vendor literature describing how Unclassified information can be securely transmitted from a user in the Secret system-high domain, to the recipient in an Unclassified domain. Generally implemented on a trusted MLS system (at some stage in the NCSC evaluation), this fictional guard facilitates the exchange of SMTP with Multimedia Internet Mail Extensions (MIME) on TCP/IP. Messages and attachments, originating on the high side, are checked for strings like "Confidential" and "Secret." If these are found, the E-mail is sent back to the originator. If not found, the message is regraded low and forwarded to the Unclassified domain. Low messages are

parsed in the same manner; if the dirty words are found, the hapless user gets to meet the site security officer up close and personal.

A "more robust" solution has been identified which makes use of (Fortezza) cryptography applied to putatively low data objects in a system high environment. One of the solution sets proposed by the Multilevel Information Systems Security Initiative Program Office, in this scenario the guard (or Secure Network Server) validates that the object is encrypted properly, and then transmits the object low for delivery. This provides a very secure channel for potentially classified information to travel to the low domain.

While both of these solutions enable the comparatively easy exchange of E-mail between the high and the low domains, these solutions do not adequately address the potential for inadvertent high information migration to the low domain. To varying degrees, the solution documentation alludes to a level of residual risk which must be assumed. Since identifying this risk as potentially unacceptable would preclude a sale, the marketers stress the operational benefits of the solution.

What Policy can Allow Automated High-Low Transfer with Acceptable Residual Risk?

There is no universally applicable answer to this question. System high output must be handled as required by the most sensitive information processed or stored on the system until it is reviewed [Com95]. The depth and breadth of an automated review (conducted by a guard by implementing filters) will be a function of several parameters, including:

Must the guard address the legitimacy of advisory markings contained within the message text. Must it counter willful efforts by a telegram author to conceal Secret information in inappropriately marked messages. Said another way, can the reviewer believe that if the message

contains the string "UNCLASSIFIED" in appropriate locations, then this accurately reflects the author's belief of the message information content? If not, manual cognizant review will be required.

Does the guard have to address covert storage channels from high to low? Must data that potentially represents encrypted Secret information be deleted from objects before they are transferred from high to low? If so, sophisticated filtering will probably be required.

Does the guard have to address covert signaling from high to low? Will it be accessed interactively by high and low users? If so, a B3 or higher system should form the basis of the guard platform.

What is the projected service level? Could the number of messages associated with this level of service be legitimately reviewed by a human? Notwithstanding the legitimate technical vulnerabilities, if the number of messages exceeds number that can be reviewed, than an admittedly risky, automated solution is required. Minimizing the technical risks becomes the goal of the guard architecture.

Cognizant humans should review an object to change its handling requirements from that required for system-high information. If the resource/manpower costs for human review of the messages are so high that this is not a viable option, an alternative strategy must be developed. Even if cost is not an issue, review quality could be a factor, since even the most conscientious reviewers eventually become fatigued.

Ideal E-mail Guard Platform

What is the ideal automated guard for the system high processing discussed herein? The filtering applications should be hosted on a trusted, multilevel secure system. The use of a trusted multilevel system allows a

reasonable level of assurance that the filtering functionality within the guard cannot be easily bypassed. Stated another way, the features of a multilevel system, especially mandatory access controls, allow a security policy to be established within the guard such that message traffic which enters the guard on the **high** side (from the Secret system—high domain) can never leave the guard on the **low** side (to the Unclassified domain) without a regrading of the message sensitivity level. This regrading will ideally occur as a result of the telegram successfully meeting the criteria of the review processes, but can also occur based on the manual review of a human user with the requisite privileges.

The ideal candidate for an E-mail guard host platform is one which offers high assurance at a low price. Since the assurance level of the TCB is proportional to the assurance that the messages were filtered prior to transmission, in some respects, the higher the assurance the better. If two domains with disparate risk environments are interconnected, then a high assurance trusted MLS solution is essential.

Guard Application Filters

The TCB will host a guard application which can be viewed as a number of filters. The guard software is designed and developed to meet specifications. One of the specifications should ensure usability. The best message guard in the world is worthless if it can only be managed and operated by computer scientists or systems programmers.

Specifications are detailed descriptions of the features required of a program that programmers write code to produce. Specifications implement requirements, which relate the features and functions that a system must have. These requirements, in turn, stem from policies, which guide the actions of the agency. Policies can be set at the national level, such as Executive Orders, or at the agency level, such as a DoD regulation. The policies upon which the guard processor

requirements are based can be summarized as, "Classified information must never be handled as if Unclassified" and "Attacks launched against the guard, especially by low-side users, should never succeed."

For the sake of discussion, three fairly typical constraints are provided for this hypothetical example of guard operations.

The messaging application has transmitted many messages correctly, i.e., in a manner which has not lead to spillage of information. Errors have, nonetheless, been observed.

The second constraint is that the manual review of messages should only occur if required by a filter, since the volume of messages exceeds five thousand per day.

The third constraint is that the guard application does not have to detect classified information willfully inserted by the drafter into a message destined to a low-side recipient.

Several approaches can be taken in developing requirements for this guard. These approaches, which are based on the functionality that the guard should implement, can be broken down into three general areas: looking for indicators of correctness in message processing, looking for indicators of mistakes in message processing; and evaluating patterns and meanings of text strings. The last general area of filter mechanism attempts to accomplish actions which are similar to those of a human reviewer.

Correctness is an attribute of processing which can be measured only if a metric for correctness exists. For messaging, the metric must be established with respect to messages transmitted from the high side to the low side, after which, the guard can examine messages to determine if they appear to be correct. These values can then be parsed, and evaluated by the guard application.

A procedural requirement should be established that messages intended for the low side contain the string "(U)" or "UN" as the first characters of the subject line in the message header.

Likewise, users should be required to begin and end the message text with the string, "Unclassified."

A recipient of the message (a name/domain pair) may not exist on the high side. Messages intended for the low side should be sent only to low-side addressees, to reduce the impact of inadvertent message transfer errors.

Similarly, filters can attempt to identify messages that are incorrect, i.e., they lack the necessary attribute of correctness. Filtering on the inverse of the last three criteria allows potentially incorrect messages to be identified. Additional criteria can also be developed which potentially indicate incorrectness such as:

A message could contain any of a number of dirty words, i.e., strings, the presence of which could indicate classified information. Such strings include: "Secret", "NOFORN", "OADR", "SECRET", "CONFIDENTIAL", etc. A capability to identify clean words, like "Secretariat", "Secretary", "LOADR" allow the filter to be more efficacious.

A message could contain attachments other than text. Since such attachments cannot be reviewed in a trustworthy manner, they are ineligible for a lesser handling requirement, and may not leave the Secret system-high domain.

Since message text cannot be modeled (i.e., message text cannot be predicted), the first two types of guard mechanisms (correctness and incorrectness attributes) have only limited utility to filtering the actual text of the message.

Nonetheless, structural characteristics of messages, if pre-ordained, can be used for developing rules for detecting some anomalies. Anomalies are elements of a message which do not meet a rule. The most important feature of an anomaly is that it could represent the spillage of information from one message into another. Human reviewers will have to determine if the anomaly represents a mistake, or if the anomaly is simply a different format from the required norm.

The rationale for focusing on structural characteristics stems from the constraints provided. Since the mechanism for message transmission and the system-high environment are viewed as benign, the focus of the filters is on identifying random, non-deterministic errors, which could result in spillage. An example of such an error is four lines from the text of one message being appended to a paragraph in a different message. These errors would be detectable by the guard only if the error causes a format rule to be violated.

Operational and security goals conflict directly when determining such rules. If a rule is violated, and the violation is not caused by a message processing error, a type 1 (or false positive) error occurs. A human is inappropriately required to review a correct message if a type 1 error occurs. Conversely, incorrect telegrams which do not violate structure rules represent type 2 errors, or false negatives. Type 2 errors occur when a human should review the message, but the automated filters did not detect an incorrect state of the message.

Operationally, minimizing type 1 errors is a significant goal, since type 1 errors require human intervention unnecessarily. From the security perspective, minimizing type 2 errors is the most significant goal. Effective rules can minimize both types of errors. Potentially, type 1 errors will increase as type 2 errors decrease in the guard processing. This

results from being able to identify anomalies only to the extent that format rules are violated. Said another way, the more precisely the message format is modeled, the more false positives (telegrams which the guard does not allow to pass, even though the text is innocuous) will be detected. The degree to which the format of message text matches a predefined pattern is a function of how well the message originators adhere to preestablished norms, hence the problem.

The third class of mechanism, which includes text and language analysis, best counter information security vulnerabilities. Because of its ability to only execute preexisting instructions, the most difficult thing for a computer guard to do is to act, or react, like a person. Human behavior (in this case, the actions a concerned, human reviewer takes while reviewing messages) is infinitely complex, and thus, almost impossible to model. Conversely, if the desired behavior can be modeled, it is normally possible to automate.

Guard Functions

Since a human reviewer would not be charged with validating the correctness of the Unclassified marking (would not have to second guess the author as to whether a message is under-classified), what would a reviewer do? This distinction is not trivial since reviewers in guarded environments, like the OIW, often **must** completely review data objects for information content before the data object is allowed to transit to the low side of the guard.

Conscientious reviewers probably scan messages, instead of reading them. While scanning messages, reviewers would be looking for any indication of potential spillage. Finding an indication (such as a subject line embedded in the second page of the message text) the reviewer would then read the text to determine if spillage had occurred. Scanning

would focus on identifying any of at least five indicators of possible spillage:

1. Is the message legible, i.e., in English? Said another way, do the words (i.e., character strings delineated by spaces and/or punctuation marks) exist in the agency's lexicon?
2. Does the message make sense? Does the syntax (the way in which words are put together to form phrases and sentences) convey a message that is understandable?
3. Does the theme of the message change? Is there an unanticipated change in the subject of the message (e.g., a paragraph about satellite perihelion within a message reporting on unit readiness in Europe)?
4. Does the message contain markings or words indicative of classified information?
5. Is the message complete, e.g., nothing extra and nothing missing?

All of these activities would occur concurrently as an operator reviewed the message. Whether or not these activities, especially items two and three, could be automated is unclear. To legitimately get the human out of the loop without a potentially unacceptable high level of residual risk, all of these activities should occur on a message guard reviewing free-form text.

Of the five activities identified, only two (items four and five) appear to be relatively easy to automate. Items one and two could potentially be automated, but the processing impact could be significant. Determining the legibility of a message (item one) can be automated using dictionary look-up techniques. In a like manner, the message syntax can be evaluated (item two) using methods similar to the Grammatik application, a PC-based grammar and syntax parser. A specialized form of such a parser has been described [Lun87] in which a rule-based expert system

uses syntax and word relationships to classify certain data items appropriately.

Evaluating for an inappropriate theme change (item three) within a message could be accomplished syntactically, keying on geographical location or noun family. Natural language analysis continues to progress (the reader is invited to listen to the marketing spiel about Oracle's® Context® application) but this type of check will possibly always require a human.

As discussed previously, message text can be scanned for specific delimited strings (item four) such as "(TS)", "CNWDI", "CLASSIFIED" which indicate the potential presence of classified information. The utility of this will always be marginal.

If message format can be relied upon as a metric for completeness, the fifth activity can also be automated. This proposal to use message format as a metric of completeness is not rigorous. Scanning for known structures, such as the presence of a period at the end of a paragraph, or the string "Unclassified" at the end of message text, provides limited assurance, at best, that incomplete messages (messages that are missing required information or contain extraneous information) are identified. Notwithstanding these weaknesses, this is the most practicable manner of enabling the fulfillment of operational requirements. While vulnerable to exploitation by Trojan horse attacks, the likelihood of inadvertent errors in a closed environment can be reduced by structural filtering.

Filter Requirements

Structural similarities in message format can be used as filters for anomaly detection, especially if a unique format is administratively required for low information-content messages. Since the format can be modeled, it can be filtered against, and thus identify

messages which should be reviewed by a human.

What should the acceptable format be?

This will be site specific, but the following illustrate some of the possibilities:

An unclassified line marking could be required for all paragraphs.

If numbers are used for paragraphs, these should be in order both serially and alphabetically (i.e., paragraph 1.A precedes paragraph 1.B., which is followed by paragraph 2, etc.)

Additionally, paragraphs should end with a period, a right parenthesis, or a colon, followed by either a CR/LF, and CR/LF/LF.

At least 99% of the delimited strings comprising the message text should appear in the corporate dictionary if the message is less than ten kilobytes in length. If the message size is greater than ten kilobytes, 99.5% of the strings making up the message text must be in the corporate dictionary.

Specific specifications for a free-form text guard with structural filtering include:

1. Spill for manual review all messages without the string "(TS)" as the first characters of the subject line.
2. Spill for manual review all high-low messages with a recipient that exists in the Secret system-high domain.
3. Spill also for manual review all low-high messages with an originator that exists in the Secret system-high domain.
4. Spill for manual review a message which contains text, the structure of which does not meet format rules.

5. Spill for manual review all messages which contain too many strings which are not in the corporate word list.

6. Spill for manual review a message which contains an occurrence of strings indicative of classified information.

7. Spill for manual review a message which contains invalid paragraph numbering. Validity is defined in the message format rules.

8. Spill for manual review a message which contains invalid paragraph ends. Validity is defined in the message format rules.

10. Spill for manual review any message that does not conclude with "Unclassified" as the final string.

Summary

Only textual information can be transmitted with minimizable security concerns between domains processing at differing system-high sensitivity levels, especially between those with different risk profiles. Lexical and syntactic parsing of message text could allow a significant increase in assurance that text transiting a guard processor from high to low does not contain classified information. The feasibility, processing impact, and increased cost from development efforts relating to such automated review make such parsing questionable. Nonetheless, free-form text messages can be filtered against administratively established structural characteristics to reduce the inadvertent migration of information in a closed networking environment.

A solution to the security vulnerabilities associated with meeting multilevel messaging requirements can only be found with the use of trusted MLS systems at the point of origin of the data. The exportability of labels from these systems allows a mandatory security policy to be enforced at the interface between

the domains without unnecessary human intervention.

Reference

- [Ber90] Jeffrey L. Berger, Jeffrey Picciotto, John P. L. Woodward, and Paul T. Cummings. Compartmented Mode Workstation: Prototype Highlights. *IEEE Transactions on Software Engineering*, 16(6):608-616, 1990.
- [Com85] Department of Defense. DoD Computer Security Center. *Computer Security Requirements -- Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*. June 1985. CSC-STD-003-85.
- [Dep85] Department of Defense. National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria*. December 1985. DoD 5200.28-STD.
- [Def94] Defense Information Systems Agency. Message Guard Assessment. *Department of Defense Multilevel Security Program Technical Memorandum*. June 1994. FOUO.
- [Kur92] Charles Kurak and John McHugh. A Cautionary Note on Image Downgrading. In *Proceedings of Eighth Annual Computer Security Applications Conference*, pages 153-159, San Antonio, TX, December, 1992.
- [Lun87] Teresa F. Lunt and Thomas A. Berson. An Expert System to Classify and Sanitize Text. In *Proceedings of AIAA/ASIS/IEEE Third Aerospace Computer Security Conference: Applying Technology to Systems*, pages 30-34, Orlando, FL, December, 1987.
- [McH85] John McHugh. An EMACS Based Downgrader for the SAT. In *Proceedings of Eighth National Computer Security Conference*, pages 133-136, October, 1985.

Definition of "guard," extracted from *Microsoft Bookshelf © 1987 - 1992 Microsoft Corp. All Rights Reserved. The American Heritage Dictionary and Electronic Thesaurus are licensed from Houghton Mifflin Company. Copyright © 1986, 1987 by Houghton Mifflin Company. All rights reserved.*

SECURING LOCAL AREA AND METROPOLITAN AREA NETWORKS : A PRACTICAL APPROACH

Prof. Vijay Varadharajan
Dept. of Computing, Univ. of Western Sydney,
Nepean, Australia. email: vijay@st.nepean.uws.edu.au

June 6, 1995

Abstract

This paper considers the design and management of security services for connectionless services in LANs and SMDS based interconnected LANs. First the security threats in such environments are briefly outlined and the types of security services and mechanisms required to counteract these threats are mentioned. The possible options for the placement of security functions in a LAN architecture are discussed. The paper then describes the design and implementation of a secure LAN prototype system. The applicability of the developed security layer to a SMDS network is analysed. Then the design of a secure SMDS system is given.

1 Introduction

There is a growing interest in the development of broadband services and networks for commercial use in both local area and wide area networks. Recently there is a real pragmatic drive for broadband services, to meet the demand for increased bandwidth for remote sites inter-connection, and for image and high speed data transfer. Broadband activity now has commercial services under a variety of titles, some using ATM techniques such as Switched Multimegabit Data Transfer (SMDS) [1] and Metropolitan Area Networks (MANs), and others such as Frame Relay [2].

It has been clearly established that there is a need for interconnection of Local Area Networks (LANs) over a wide area. The interconnection of LANs providing high speed information transfer is becoming a strategic necessity for many enterprises to support their growing number of workgroup-based and backbone-type LANs. As the importance of IT and global communications to an organization increases, security and privacy issues are playing an increasingly significant role. In particular, with the increasing use of broadband networks and the growth in the number of applications requiring broadband services, network security issues are becoming critical to business and industry. In the past the limitation to local site operation has provided some security features, but the use of external networks and contractors, introduces new and greater risks.

This paper addresses the design and management of security services in local area network and wide area networks comprising interconnected LANs via SMDS. SMDS and Frame relay represent the best known ways of accessing multi-megabit backbones at present. We considered security for connection-oriented Frame Relay service in [3]. A paper contrasting SMDS and Frame Relay solutions is currently in preparation. In this paper, we consider security for connectionless services. The paper is organized as follows. Section 2 summarizes the security threats in a network environment and outlines the types of security services and mechanisms the network architecture needs to support to counteract these threats. Sections 3 and 4 address the provision and management of security services in a local area network. Section 5 extends this to a SMDS wide area network and describes a secure SMDS service. Finally Section 6 provides a brief summary.

2 Networked System Security

A typical networked system environment comprises networks, users (people), information storage resources, computing resources, and peripherals. One may have various levels of interaction and various degrees of sharing between these entities. At one level, we may have interaction between different resources: such as host to host, host to peripheral, and host to storage resource. At another level, one may have interactions between applications in different entities. At the outer level, we can have interactions between users. Furthermore, the degree of interaction and sharing may also vary. At one level, there may be just transfer of information (e.g. email, edi). At another level, one may have sharing of computing resources (e.g. processors) and sharing of information storage resources (e.g. disks). At the next level, one entity (e.g. an object) may act upon another entity (e.g. object) to obtain a service from the latter. A still greater degree of cooperation occurs when entities jointly work together to perform tasks. In each of these activities various attacks can occur and hence the need for security measures become significant. Furthermore, in a wide area network, it is important to consider the organizational structure and boundaries. This has impact on who has responsibility and authority for which parts of the network and for what services. This is crucial for determining the placement and operation of security functions, and the interactions between them.

In addressing the overall security of a network system, it is necessary to integrate computer system security (COMPUSEC) and communication security (COMSEC) measures to protect information both *within the system* and *between systems*. Neither one on its own can provide the required complete protection of information in a networked environment. For instance, access controls to restrict users gaining access to a resource within a system or on a network together with suitable flow constraints to regulate the flow of information are essential. Trusted computer system mechanisms are needed to ensure the enforcement of security controls and in the provision of the necessary assurance that the correct operation of the security measures are maintained. Secure protocols are vital to the successful operation of security measures. Security mechanisms like encryption algorithms form an essential part of the overall solution.

2.1 Security Threats and Services

Unauthorized disclosure of information via eavesdropping and wiretapping is perhaps the most common threat that comes to one's mind when one thinks about network security attacks. Confidentiality service is used to protect information from unauthorized disclosure. In a network environment, often it is important that such a service is provided in an end-to-end manner, thereby ensuring that the information is protected over the complete network path.

In a *masquerading attack* one entity pretends to be an other and attempts to gain privileges and access to information and resources to which it is not authorized. User and origin authentication services can be used to counteract such attacks. Mechanisms used to realize this service include the use of challenge-response and cryptographic techniques in the implementation of secure authentication protocols.

Another common attack is the *unauthorized access* to network resources. This can involve network components such as printers or network resources such as operating systems, databases and applications. Access control service provides protection against unauthorized access and use of resources in a network system.

The threat of *unauthorized modification* of information and resources causes integrity violation. Such an attack may involve unauthorized insertion and deletion of information transferred over the network. This attack often occurs in conjunction with other attacks such as replay whereby a message or part of a message is repeated intentionally to produce an unauthorized effect. Integrity service provides for the protection of information from unauthorized modification.

Repudiation of actions is another form of attack that can occur in a networked system. It occurs when a sender (or a receiver) of a message denies having sent (or received) the information. The non-repudiation security service that can be used to counteract such a threat provides proof of the

origin or delivery of information. Provision of such a service requires some form of digital signature mechanism. Such a service also implies the existence of an agreed trusted third party whose primary role is to arbitrate disputes resulting from non-repudiation.

Unauthorized denial of service attack by an entity involves the denial of a service to another entity even though the latter is authorized to access that service. That is, an entity prevents other entities from carrying out their legitimate functions. In a network, this form of attack may involve blocking the access to the network by continuous deletion or generation of messages so that the target is either depleted or saturated with meaningless messages. Denial of a service can be regarded as an extreme case of information modification in which the information transfer is either blocked or drastically delayed. The measures provided by confidentiality, integrity and authentication services can be used to detect some, but not all, forms of denial-of-message-service attacks. For instance, they cannot detect such attacks if they begin while the communication association is quiescent. In such a situation, the receiving entity has no way of determining when the next information should arrive. In fact, it will remain unaware of the attack until it attempts to send information itself. In many cases, the entity attempting to send the information will detect the attack but it has no way of notifying the other entity. A measure against such an attack is to have periodic exchange of information between entities to ensure that an open path exists between them. The greater the frequency of such a request response mechanism, the shorter the time period during which the denial-of-service attack will remain undetected. However the disadvantage is that this reduces the effective bandwidth of the network. The security audit service is somewhat orthogonal to all the security services described above in that it is not directly involved in the prevention of security violations but assists in their detection.

2.2 Security Management

Key to provision of a security service is its management. A network security architecture needs to support the management of these services and how changes in policy and its enforcement can take place. For instance, in the case of confidentiality and integrity services, it is necessary to manage the keys used in the encryption and decryption processes. In the case of access control service, we need to manage the access control information such as access control lists and the access rules. Similarly in the case of authentication, authentication information, e.g. passwords and keys, needs to be managed. In the case of auditing, the management of audit trails and audit analysis is necessary.

In networked systems, it is likely that there is no single authority that controls the entire environment. For instance, in an organization there may be several security managers responsible for a subset of users, objects and operations. This does not mean that it is not possible to control security in a distributed environment centrally. However even central security authorities end up trusting that the authorities responsible for local systems have implemented appropriate security. There may be several authorities performing different aspects of these security management functions : access control authorities, authentication authorities, key management authorities and audit management authorities. In practice several of these functions may be handled by a single authority.

3 LAN Security

LAN has certain specific characteristics which makes the need for security measures even more significant. For instance, typically data is transmitted on a media that is shared by every attached system. Therefore any system attached to a LAN can transmit to any other system on the LAN. Similarly information can be intercepted by any attached system. Even worse, a system can change the information in a Protocol Data Unit (PDU) before it is received at the intended destination. This is especially significant in LANs employing a ring topology, where every system must receive and retransmit every PDU in order for the LAN to function properly. Also because every PDU is transmitted to every other system on the LAN, it is difficult to identify the source of a given data transmission. Hence one system can claim to be another system. Hence the threats of masquerading, unauthorized disclosure and modification are further aggravated in a LAN environment.

Therefore it is clear that there is a need for security measures in a LAN environment. The question now arises as to where in the protocol stack should the security services be provided. For the application information to be protected, it is important that certain security services need to be provided at the application layer. This will offer protection at the highest possible level in the stack. In addition to this, there may be a case for providing protection at a lower level in the stack. The higher layer security service will not protect the header information of the lower layers, as only the service data unit of the higher layers's PDU is protected. So for instance if data integrity of header information is required, then a data integrity service at the appropriate lower layer will be required. Also there may be PDUs that originate and terminate at the lower layers, which will not be protected by a higher layer security service. Perhaps the most compelling reason may be that the security services at a lower layer offers a uniform, common protection for all applications from threats that are intrinsic to LANs. In this sense, security at a lower layer such as the data link layer can be seen as a "door lock" which acts as a first barrier, irrespective of any additional security that may or may not be provided at the application level.

3.1 Security Layer for LAN

Let us now consider the possible options for the placement of security within a LAN architecture. Security can be provided at the Medium Access Control (MAC) layer, or in between the MAC and the Logical Link Control (LLC) layer, or at the LLC layer or above the LLC layer.

- The integration of security within the MAC layer will impact a number of existing products and established standards. Furthermore, given that a common interface for different MAC standards is emerging, it is logical not to integrate security within the MAC layer but to place it above it. However note that with such an approach, it is not possible to address the threat of traffic flow analysis in an adequate manner.
- The simplicity of the interface and the protocol is the major attraction for placing the security layer between the MAC and the LLC sublayers. The security layer can be made transparent to the MAC and LLC layers with no changes to their external interfaces.
- Integration into the LLC provides several advantages. It provides the possibility for providing security services for both connectionless-mode and connection-oriented service operations. The connection-oriented mode is not possible if the security layer is placed between MAC and LLC. Connection-oriented mode does offer certain advantages with respect to key management. The first is that the key granularity can be based upon the connection, and not simply between the two peer entities which would be the case in the connectionless operation. This would allow different keys to be used for different connections, providing better security in some cases. Also the protocol can discard the key after a connection terminates, as the key is based on the connection. The connectionless scheme would require a cache to store all recently used keys as there is no concept of connection. Some other criteria would be required to discard and replace keys.

The second advantage arises due to the fact that most encryption algorithms use some form of chaining to counteract the dictionary attack which makes use of repetitions in the ciphertext. With the connection-oriented service, cryptographic chaining of multiple PDUs can be provided, thereby reducing the overhead of re-initializing the cryptographic algorithm after each PDU.

Furthermore, connection-oriented integrity service is distinctly a different service compared its connectionless counterpart, in that it ensures that the data units arrive in sequence and that all the data units over the connection have arrived. Note that the connectionless integrity is only concerned with integrity of a single PDU. The provision of the connection-oriented integrity in LLC is somewhat similar to providing a connection-oriented LLC over a connectionless-integrity layer between LLC and MAC. In this case, the sequence numbers as well as the data would

be protected against modification. However a problem would remain in the case of integrity protection against truncation.

Finally, with the security integrated into the LLC, the granularity of security decisions and enforcements can be at the granularity of the Link Service Access Points (LSAP) instead of MAC addresses.

Hence there are several reasons why the provision of security within the LLC layer may be attractive. The major issue however is the need to change the implementations of the existing LLCs. The connection-oriented security services would require changing the way that the PDUs are processed. The existing equipment would need to be modified and migration to a secure version made more difficult. In fact, it is for this reason that the IEEE 802.10 Standards group has chosen to recommend the placement of security services in between the MAC and the LLC layers.

3.2 Secure Data Exchange Layer

The IEEE 802.10 group has introduced a security layer called the Secure Data Exchange Layer (SDE) between the LLC and the MAC layers. It is argued that a LAN layer 2 exhibits similarities to a Wide Area Network (WAN) layer 3 [4], and should therefore provide the services recommended by ISO for layer 3. To counteract the LAN security threats mentioned above, one would need at least the following security services : access control, data origin authentication, data integrity and confidentiality. The SDE layer is completely transparent to the surrounding layers and no changes are required to their internal protocol mechanisms or external interfaces. The transparency of the SDE layer is intended to make secure systems compatible with non-secure systems.

A Secure Management Information Base (SMIB) can be used to store security attributes for each association maintained within the LAN. It is part of the Management Information Base (MIB). The attributes include security keys, flags and identifiers needed by the SDE protocol in the implementation of the security mechanisms. The SMIB can be implemented as a table of entries, one for each communicating pair of hosts. It allows the security management applications to control the operation of the SDE layer. There are three types of objects that can be managed : end system, SAP and association objects. The system objects are global and apply to the entire SDE regardless of the security associations. The SAP objects are applied to an SDE SAP, and the association objects only apply to a particular association.

The SDE layer implements security by manipulating the transmitted packets on the LLC-MAC layer boundary according to the security attributes stored in the SMIB. These manipulations are completely transparent to the upper protocol layers in that neither the format nor the semantics of the packet are modified. Any LLC packet passing through the SDE layer is protected and passed onto the MAC layer. Similarly, the received packets pass through the SDE layer before entering the LLC layer.

4 A LAN Security System

The provision of security services is dependent upon the perceived security threats and the trust assumptions made about the LAN environment. Consider for instance a backbone LAN connected to several sub-LANs. This is a typical configuration in large organizations who tend to organize their LAN in subnetworks which form logically connected groups, e.g. on a departmental basis. In general, we may have WANs interconnecting the LANs.

Case 1 : It may be that the intra-departmental traffic is trusted while inter-departmental communication needs to be protected. If this is the case, then it may be adequate to provide subnetwork to subnetwork security. This can be achieved by implementing the security layer in an internetworking component such as a bridge or a router. These secure bridges and routers effectively act as firewalls. The data transmitted by an end system is processed by the security layer of the local bridge or the

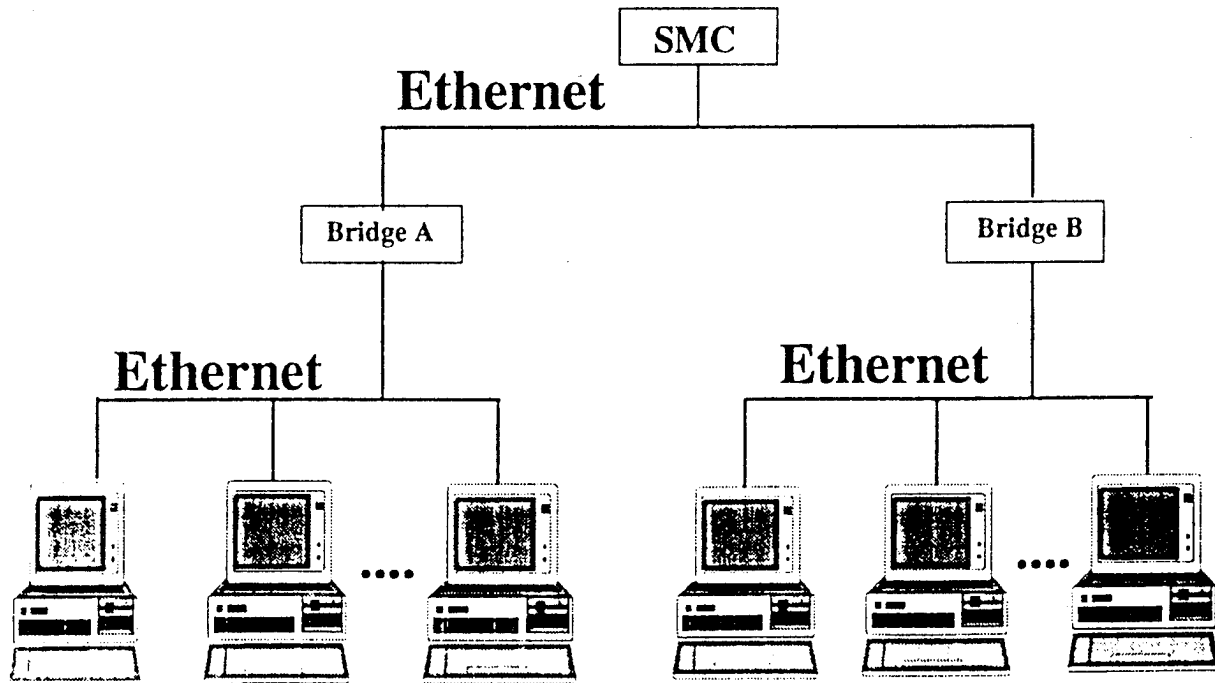
router. On reception, the remote bridge or router unwraps the security functions and passes the information to its end systems over its local LAN. Hence the information is only protected between the internetworking components. Such an approach gives an easier migration path from a non-secure LAN to a secure LAN.

Case 2 : If the local LANs within the subnetwork are not trusted, then the security services must be implemented at the end systems themselves. In this case, every time an application entity sends data to another remote application entity, the data has to be pass through the security layer. The information remains protected until it is processed again by the security layer of the remote end system.

4.1 A Secure LAN Prototype

Let us now briefly consider a protoype LAN security system that we have designed and implemented¹. The LAN configuration is as shown in Figure 1. The bus/tree topology has been adopted for the prototype, using as branches the bridges and as leaves the end systems of the network. Local LANs are assumed to be trusted and the objective is to protect the information over the backbone LAN. This corresponds to Case 1 above. Hence the security layer was implemented at the bridges and not at the end systems in this prototype. Each bridge provided security services to the end systems in its subnet, when they establish an association with the end systems in other subnets. The security layer offered the all the SDE (IEEE 802.10) security services.

Figure 1 : Secure LAN Prototype Configuration



The prototype system comprised two Secure Bridges (SB) supporting a subnet each and a Security Management Centre (SMC). The SMC is involved in activities such as the initialization and updating of information in the secure bridges, the authentication between the secure bridges, and the addition and removal of end systems in the network. In general, we may have one such Security Management Centre per backbone LAN. This would then require peer-to-peer protocols between the SMCs.

¹The author headed the Distributed Systems Security Group at HPLabs.,U.K., where this prototype was designed and built.

The security attributes such as secure associations and keys are stored in Secure Management Information Bases (SMIBs). The SMC initializes and updates security attributes in a secure way. The network configurations and the associated security policy are specified in configuration files. The SMC has a user interface for the Network Security Manager (NSM) to control the security policy. For instance, using this interface, the NSM can inform the SMC and the SBs of the configuration of the LAN, manage the LAN access control policy, determine the status of the SBs on the LAN, and control network security auditing.

4.1.1 Security Management Components and Interfaces

The prototype was supported by SNMP based messaging thereby allowing the NSM to set up and delete secure associations, manage router and end system configurations, and key exchanges. It had the following management components and interfaces (See Figure 2).

- A Key Management Process (KMP) which provides mechanisms and services to execute a distributed key exchange for a security association.
- Security Management Application Entities (SDE-AE and KMAE) : These customize the SNMP GET and SET client primitives. Each of these ask the SNMP Manager to issue an SNMP message activating one of the possible remote management operations. The SDE-AE performs creation, removal and display of a security association, configuration of security attributes (router and end systems). A Key Management Application Entity (KMAE) generates each message belonging to the KMP. It requests the SNMP Manager to activate a distributed key distribution session, assigning new keys to the selected association.
- Security Management Centre Interface (SMC-IF) provides the network system manager with an interface for executing the security management operations.
- SMC-SMIB has two logical components : SDE-SMIB and KMP-SMIB. It stores security management information such as network addresses, the security association map of the network and information about the status of local SDE-SMIBs in the bridges, as well as protected files storing keys that are used to encrypt and decrypt management messages to and from bridges.
- The SDE-SMIB and KMP-SMIB parts of the SMIB support the security architectural model by operating interfaces between management applications, such as the SMIB-HANDLER and the KMAE, and the operational layers KMP and SDE. A SMIB-HANDLER responsible for writing and reading security objects stored in parts of SMC-SMIB.
- SNMP Manager implements the protocol mechanisms that assure the reliable transmission of SNMP application messages over the end-to-end transport protocol.
- SNMP Agent responsible for the controlled reception of SNMP application messages from the SNMP Manager of the bridges.
- SNMP Handler is an SNMP agent application managing the key management objects in the KMP-SMIB after a key management message has been processed successfully by the Agent.

A Secure bridge has the following management components :

- KMAE to handle a distributed key management session.
- SNMP Manager : responsible for the reliable transmission of key distribution messages generated by the KMAE.

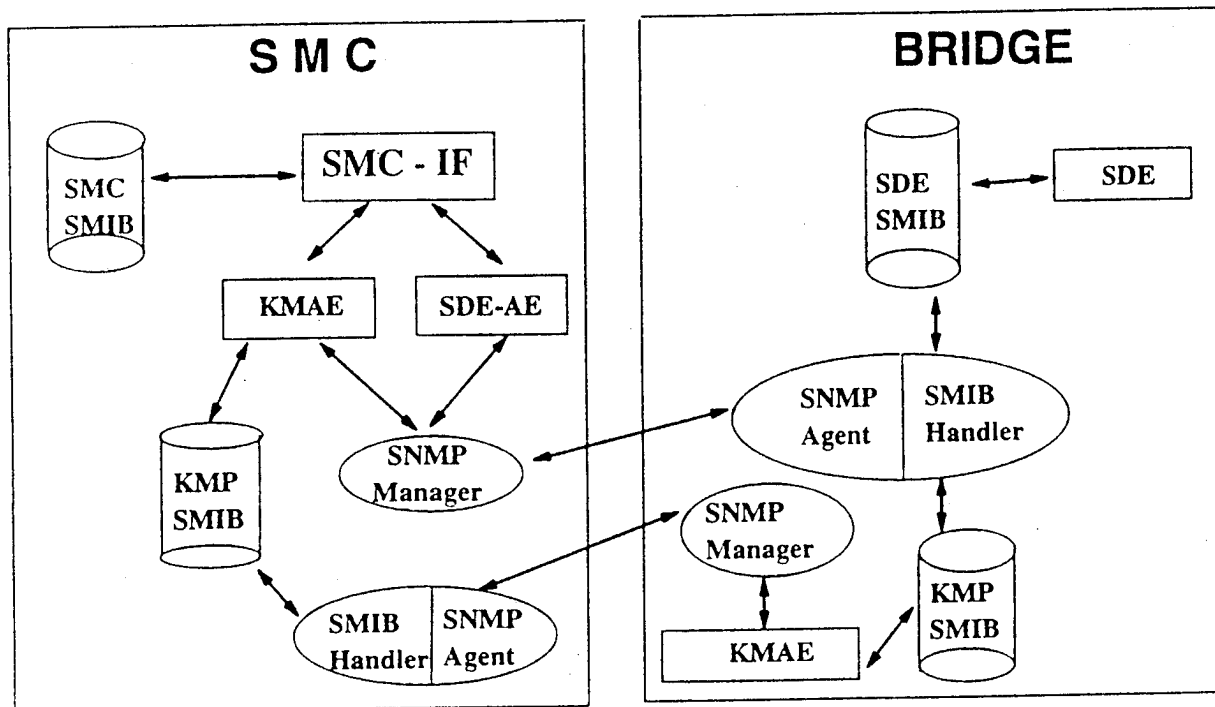
- SNMP Handler : On the receiving side manages both the KMP-SMIB and the SDE-SMIB after a successful SNMP message. The SNMP Handler reads and writes all station and security association objects to the SDE-SMIB as requested by the Network Security Manager of the SMC.
- Local SMIB (LSMIB) has two logical components, namely KM-SMIB and SDE-SMIB. KM-SMIB is similar to the KMP-SMIB in the SMC. SDE-SMIB contains security objects characterizing the secure bridge as well as the security associations between bridges.
- SDE layer performs the security controls and operations on transferred packets which are determined by the security attributes contained in the LSMIB.

Initialization Procedure

The SMIB at the SMC segmented by the KMP, creating one LSMIB for each bridge on the network. LSMIB of the bridge contains all the security attributes requested by the SDE layer of the bridge, in order to support the security services for the end systems in its subnet. Each LSMIB is encrypted by the KMP under the master key of the bridge to which it belongs and is stored in a file. When a bridge is ready to join the network this file is read by the bridge as part of the installation procedure. The network system manager NSM can change the security policy by modifying the contents of the configuration files and reactivating the SMC. The LSMIBs in the bridges will then be suitably updated.

The key management protocols designed for the LAN Security prototype are also applicable to SMDS Security discussed in Section 5. They are described in a separate paper which is in preparation. (If required, this could be provided in an Appendix.).

Figure 2 : Security Management Components

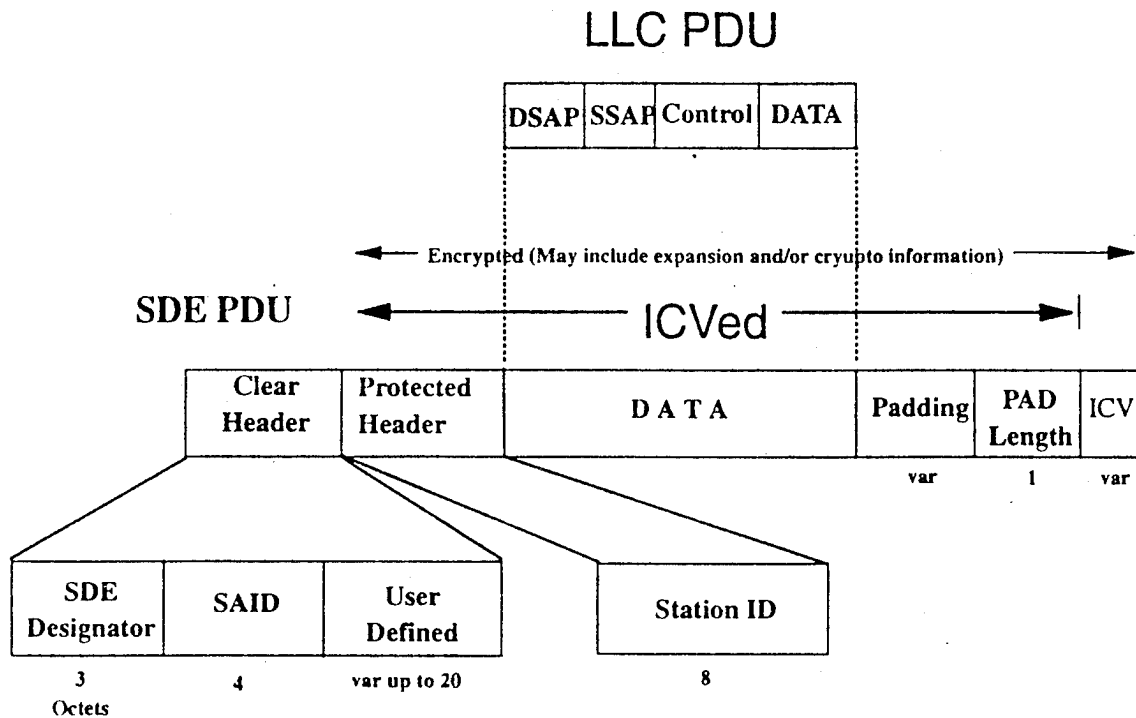


4.2 Security Services

Before considering the security services offered by the LAN system, it is important to note that the system has three types of keys, namely Secure Bridge (SB) Master Key (SBK), SMC Master Key

(SMK), and Session Key (SK). Each SBK is unique and is only known to the SB concerned and its SMC. This key is used in the communication between the SB and the SMC. SMK is used to encrypt the master keys of the SBs that are stored at the SMC. Encrypting this information prevents an attacker from compromising the security of the LAN by obtaining the copy of the information stored at the SMC. SMK is only known to the SMC and the Network Security Manager (NSM). Session Keys (SKs) are used to encrypt communications between pairs of nodes which are protected by the SBs. A different session key is used for each pair of nodes which communicate. Session keys are distributed by the SMC and are only known to the SMC and the SBs involved in the communication. The session key is unique with respect to a given session. The key management protocols in Section 6 will describe how these keys are distributed between the communicating entities.

Figure 3 : SDE Packet Structure



The following security services are offered by the LAN security system

- **Access Control** : The access policy determines whether a system N1 connected to a secure bridge B1 can communicate with a system N2 connected to a secure bridge B2. It is the system management's responsibility to set up associations and the security layer's responsibility is to enforce the access control policy. The system management establishes which associations are valid according to security policy and these are specified in the SMIB. This is then used by the key management process in determining whether an association between certain specific entities is allowed to have appropriate keys thereby providing access control. Initialization and modification of access control information are done by interacting with the SMC. The communications between the SB and the SMC are protected using SB's key.
- **Authentication** : The session keys shared by the SBs are used in authenticating one bridge to another. Key updating is done by interacting with the SMC. In particular, if a bridge does not have a shared key for another bridge, then it requests the SMC to establish such a key. The SMC first checks its access control information to decide whether such a communication is allowed. If so, it generates the keys and distributes them to the source and the destination bridges. There may be cases where the SMC may decide it is necessary to change or revoke the keys for a particular pair of bridges. In this case, the SMC will initiate the key change procedure. The communications between the bridges and the SMC are protected using keys of the SBs.

- **Confidentiality** : Confidentiality service is provided using encryption mechanisms implemented in the bridge. In general, one can use either a symmetric key or a public key approach. From a computational point of view, a symmetric key based system is more suitable for data transfer, and hence this has been used. The prototype implemented the DES algorithm. Each bridge's LSMIB contains a list of symmetric keys of all the other bridges and the SMC. These keys are used to provide connectionless data confidentiality between two bridges and between a bridge and the SMC, by encrypting the LLC PDU under the appropriate session key (See Figure 3).
- **Integrity** : Once again, a symmetric key based scheme has been used to provide this service. The packet is first hashed and then the resulting hashed (smaller) packet is encrypted using the receiver's key (e.g. the key of the receiving node or the Security Centre) to produce a checksum ICV. The ICV is appended to the PDU and is integrity protected by encrypting it under the session key of the association (See Figure 3). The bridge or the SMC at the receiving end can then calculate the checksum and see if it matches with the received one. This would not only ensure message integrity but also provide origin authentication as the sending end system's key has been used. A public key approach, would provide non-repudiation in addition to message origin authentication and integrity. In practice, other parameters within the message such as a sequence number are included to provide timeliness.
- **Auditing** : Auditing of packets and associated information such as the source and destination end systems are done by the bridge. The security audit data can then be transferred to the SMC which then analyzes the data and takes appropriate actions.

5 SMDS Security

5.1 SMDS

SMDS is a connectionless public packet switched service which is currently defined to operate at 1.544 Mbps (DS1) or 44.736 Mbps (DS3). Access to a network supporting SMDS is through a Subscriber Network Interface (SNI) over which operates the SMDS Interface Protocol (SIP). The SIP is currently based on the connectionless data part of the Distributed Queue Dual Bus (DQDB) MAC protocol defined in the IEEE 802.6 Standard.

The philosophy behind SMDS is that it should be a transparent LAN extender in that it provides "LAN type" performance over wider geographical areas either through LAN/SMDS routers or direct connection. The SIP provides a connectionless service by creating a SIP level 3 Packet Data Unit (PDU) which contains the datagram to be transported and a MAC layer source and destination address. SIP level 3 PDUs are segmented into 53 byte SIP level 2 PDUs which are sent through the network supporting SMDS to an SMDS end point. At the SMDS destination, SIP level 2 PDUs are reassembled into SIP level 3 PDUs, from which the data is extracted and forwarded to upper layer protocols.

SMDS provides an address filtering mechanism which offers a limited security service, allowing customers to subscribe to a set of addresses from which messages can be received and sent. At subscription time, an access class will be chosen by the subscriber depending on perceived requirements. This provides a lower average bandwidth than the full DS3 at a lower cost. This feature is used to control the amount of bandwidth which subscribers have access to.

5.2 SMDS Security

The communication environment supported by the SMDS is similar to the LAN's layer 2 environment. If we compare an SMDS network to a LAN, then every Customer Premise Equipment (CPE) can be thought to be equivalent to a LAN host, having its own access path to the common transfer media. Each CPE is uniquely identified on the SMDS network by an E164 address. just like each host has a

unique MAC address on a LAN. In fact, the security problems associated with the SMDS network are very similar to the ones that arise on a LAN (layer 2).

At the first instance, we considered the provision of the four security services confidentiality, connectionless integrity, authentication and access control. Since security services provide different features depending on the level at which they are implemented, the first issue is to decide the level at which the security layer must be provided in the SMDS interface. Drawing on the similarities between SMDS and LAN, one approach could be to place the SMDS security layer between the SIP and the IP layers. This provides security services over the SNI-to-SNI path, and can be applied to an end system, router or bridge accessing the SMDS through SMDS intermediary addresses.

Figure 4 shows an example of an internetworking scenario. On the one side, we have a LAN host (system A) accessing the SMDS network through an SMDS router which interconnects to a CPE (system B). The security layer can reside on top of the SIP, on both the interfaces accessing the SMDS network which in turn interfaces directly to the IP layer. This is just one possible scenario. This approach can be extended to any SMDS configuration.

5.3 SMDS Security Prototype based on SDE

Given the similarities in security requirements between SMDS and LANs, we decided to apply the SDE layer to SMDS. We developed a prototype system with the following configuration (See Figure 5): two SMDS CPEs comprising end systems B and C, an SMDS router connecting the SMDS network to a LAN and another end system A. The end systems A, B and C communicate with each other via the SMDS network. A SMC residing at the LAN manages the secure communication between the router and the SMDS CPEs. The SDE layer has been embedded in the SNIs of all the devices accessing the SMDS - router, end systems B and C. On these interfaces, the security layer can be either active or partially active or inactive, depending on the customers requirements. For example, system B may require all the four security services, whereas end system C may only need a subset of them, or even no security at all. Moreover, each subscriber may require different security service sets for its communicating entities.

Once again all the security information required by the SDE protocol is stored in the SMIBs of the router and the end systems. All local SMIBs are managed by the SMC in a manner similar to that described earlier.

While the control of security associations on which the security layer is based is devolved to the manager station, individual entities do have the right to activate and deactivate the security layer. In this way, a degree of flexibility is provided without compromising the privacy, consistency and correctness of the security attributes.

When for instance system B and the router ask for a secure communication path, the SMC sets the attributes according to the request updates both SMIBs with the new association entry. The two entities can then start communicating in a secure way over the SMDS public network by activating the SDE layer which is embedded in their SNIs. The private keys shared by these entities are protected and stored in the SMIB entry, and are managed by the SMC. When necessary, the SMC generates new private session keys for the communicating pair, distributes them to the requesting entities in a protected way (by applying cryptographic mechanisms) and updates the keys currently in use (stored in the SMIB). Mechanisms allowing synchronization of the new key between the two entities is also provided.

5.3.1 Security Management Components and Interfaces

The SMC components and interfaces are exactly the same as those described in Section 4.1.1 for the LAN Security system. Management components of the secure SMDS router/end system are very similar to those in the secure bridge. These comprise the following :

Figure 4 : SDE in an Internetworking Environment

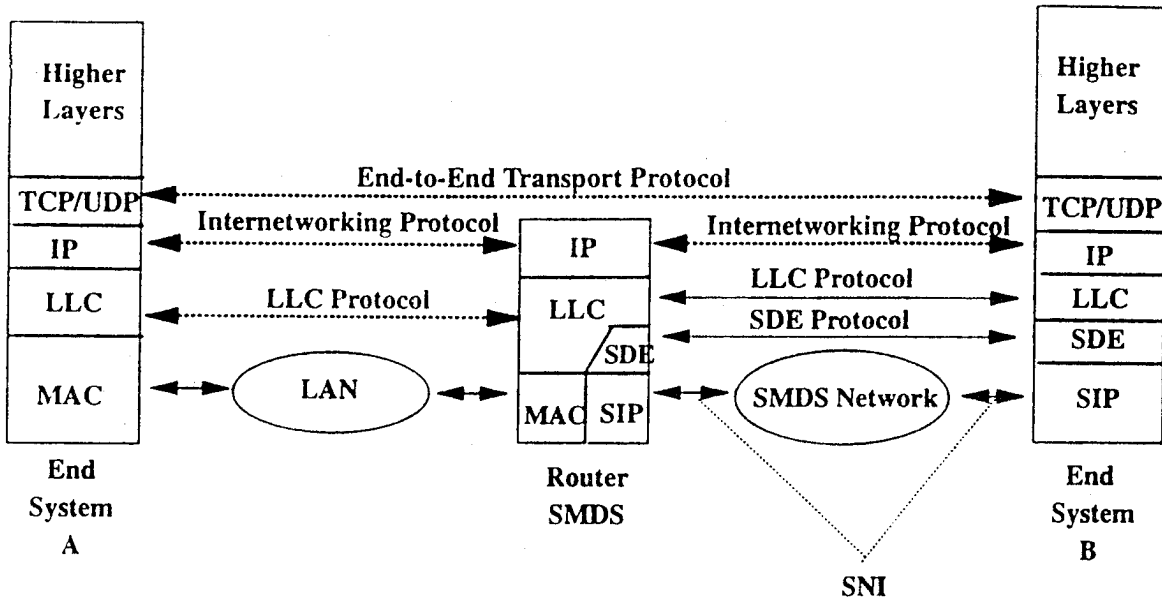
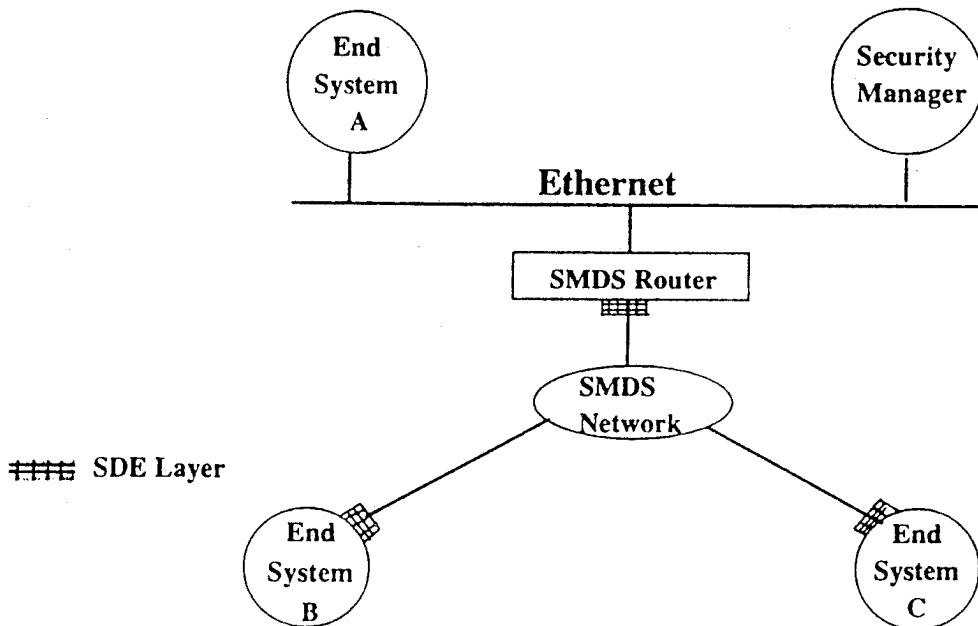


Figure 5 : Secure SMDS Prototype



- **KMAE** : It handles the distributed key management session.
- **SNMP Manager**: It is responsible for the reliable transmission of key distribution messages generated by the KMAE.

- **SNMP Handler:** On the receiving side it manages both the KMP-SMIB and the SDE-SMIB after a successful SNMP message. The SNMP Handler reads and writes all station and security association objects to the SDE-SMIB as requested by the Network Security Manager of the SMC.
- **Local SMIB :** It has two logical components, namely KM-SMIB and SDE-SMIB. KM-SMIB is similar to the KMP-SMIB in the SMC. SDE-SMIB contains security objects characterizing the secure SMDS router/end system as well as the security associations in which the end system or the CPEs attached to the router are involved.
- **SDE layer:** It performs the security controls and operations on transferred packets which are determined by the security attributes contained in the LSMIB.

6 Summary

In this paper, we have considered the design and management of security services for connectionless services in LANs and SMDS based interconnected LANs. First the paper described the security threats in such environments and outlined the types of security services and mechanisms required to counteract these threats. It discussed the possible options for the placement of security functions in a LAN architecture. The paper then considered the design and implementation of a secure LAN prototype. The applicability of the developed security layer to a SMDS network was discussed and the provision of such a security layer in secure SMDS system was described.

The key management protocol that was designed and implemented in the LAN and SMDS security system prototypes is described in another paper which is in preparation. The protocol is based on symmetric key cryptosystems. It is also possible to use public key based key management protocols. We have analysed these protocols using an extended BAN logic [6]

Acknowledgements : The author would like to thank his colleagues at Hewlett-Packard who have been involved in the implementation of the systems described in this paper. This includes Jackie Balfour, Paola Fulchignoni, Panos Katsavos, Giovanni Marotta and Tony Wiley. The author would also like to thank the anonymous referees for their valuable comments.

References

- [1] Bellcore : "Generic Security Requirements in support of Switched Multi-Megabit Data Service", Bellcore Technical Reference TR-TSV-000772, May 1991.
- [2] IETF : Frame Relay Specification with Extensions, Internet RFC 1294.
- [3] Vijay Varadharajan, Panos Katsavos, "Security for Frame Relay", Proc of the International Conference on Computer Communications, 1993, USA.
- [4] International Organization for Standardization (ISO), ISO 7498 : Part 2 - Information Processing Systems - Open System Interconnection - Basic Reference Model - Security Architecture, 1988.
- [5] IEEE 802.10, Standard for Interoperable Local Area Network Security (SILS) : Secure Data Exchange Layer (SDE), 1991.
- [6] Mike Burrows, Martin Abadi, Roger Needham, "A Logic of Authentication" ACM Operating Systems Review, 23(5), Dec.1989.
- [7] Vijay Varadharajan, Claudio Calvelli, "An Analysis of a Key Distribution Protocol for the Secure LAN and SMDS Prototypes", In preparation.

USING NETWORK TRAFFIC ANALYSIS AS A SECURITY TOOL

Peter Troxell, Curry Bartlett, Nicholas Gill

Digital Equipment Corporation

1430 Oak Court

Beavercreek, OH 45430

Contract

This work was partially funded under contract number F33600-92-D-0132 by the Air Force Materials Command 88th Communications Group and the Air Force Information Warfare Center. The viewpoints expressed here are of the authors and not necessarily those of the government.

Abstract

This paper presents a method of protecting an organization's computers by monitoring the network traffic. By then performing a traffic analysis on the connects between systems, security personnel can detect questionable activities for further follow-up. This methodology is designed to be complimentary to the use of network firewalls since it is analyzing *authorized* traffic for unauthorized content. The Network Security Analysis Tool (NSAT) was developed for the U. S. Air Force by Digital Equipment Corporation to perform traffic analysis as part of a centralized security administration environment. NSAT and its use in an operational environment will be the basis for this paper.

Introduction

One of the key mechanisms for protecting an organization's computer resources from unauthorized users is the notion of a network firewall. The primary concept of a firewall is to stop intruders at the front door by mediating what traffic is allowed to pass into the organization protected by the firewall. Likewise, the firewall controls what information can pass from within the organization to the external network. There remains a basic problem with firewalls: regardless of how they are setup, some traffic (i.e. mail) flows between the internal network and the external network. While most of this traffic is innocuous, some of it can contain material that is not suitable for transmission into or out of the organization. This material could be trade secrets being sent outside of the company, or attempted intrusions from hitherto safe hosts.

To counter the threat of unauthorized messages, the network traffic could be monitored to determine suspicious activities that require further investigation. This monitoring consists of keeping track of which systems are talking to each other and what ports the communication took place on. In addition, timestamps of the starting and ending times of the transmission along with the number of bytes sent are stored. This information can then be used to check if there has been

any other communications between these systems (historical perspective) or to see if there is a pattern of similar connections indicating some type of probing. In either case the system addresses and timestamps will allow further investigation by using the audit and other system information at the end node.

An example of how this would work in a non-firewall configuration follows:

1. Site policy states that the computers are to be used for official purposes only, i.e. no games.
2. The traffic analysis indicates that a large number of connects from outside the company are being made to port 4000 on machine A.
3. Telnetting to port 4000 on machine A gives you a welcome message for a Multi-User Dungeon (MUD) game.
4. Since MUD games clearly violate the organization's usage policy, the system manager is located and the game shut down. Disciplinary actions could then be initiated as necessary.

NSAT

As part of the Security Tools Enhancement Project (STEP), sponsored by the Air Force Materials Command 88th Communications Group and the Air Force Information Warfare Center, the Network Security Analysis Tool (NSAT) was developed to collect network packets and analyze them in a security context. The analysis can consist of either performing traffic analysis on the packets collected or session reconstruction. Using traffic analysis you can determine which machines is talking to each other, what ports are being used by each, and the length of the session in time and number of packets.

NSAT runs on Digital Equipment Corporation's VAX and AXP processors running the OpenVMS operating system. Two computers are used to support the NSAT network configuration as seen in Figure 1. The first computer (NSAT Collector) supports collection of the network packets which are periodically passed to the second computer (NSAT Processor) for analysis and storage. NSAT supports the collection and analysis based on either the DECnet, LAT, or TCP/IP address, the protocol used, or the fact that the packet contained a certain character string.

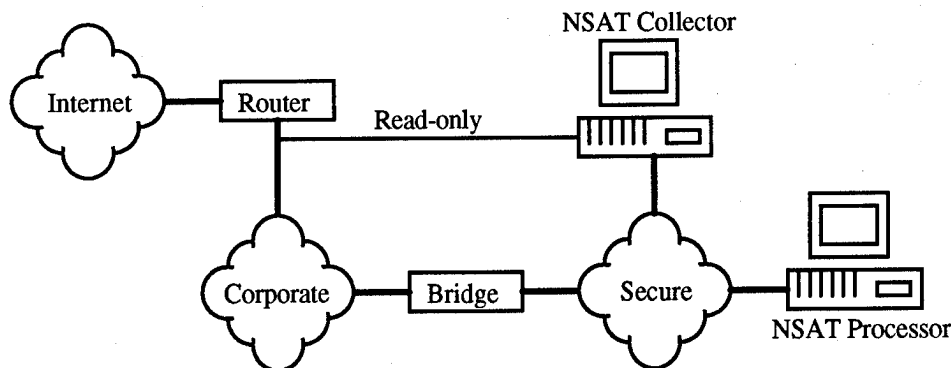


Figure 1: NSAT Network Configuration

Packet collection can be either the entire packet for cases where you want to do session reconstruction or just the protocol headers (no body) when you want to perform traffic analysis and do not need the contents.

Privacy Concerns

One of the design issues was to insure that the privacy of the network citizens would not be violated. While performing the traffic analysis, only the packet headers are retained in the collection file. These headers are retained long enough to build a connection record that details the length of the session to include wall clock time and number of packets. After processing into the connection records the packet header file is deleted.

Although recently resolved, using the keyword search capability did raise concerns centering around privacy: the keyword search program is effectively reading the contents of *every* packet on the network. Even though an individual would not be viewing the contents of any packets except those that contained the keyword, we were concerned with a possible invasion of privacy. Recent legislation now permits system managers to perform this type of monitoring. As an additional measure, all systems are recommended to have a banner that indicates that users are subject to monitoring.

Another concern was that the software would be obtained by unauthorized parties and used against others. To counter this threat NSAT makes use of Digital's License Management Facility (LMF). LMF is a mechanism in which software is not allowed to execute unless it has a valid license on file. The use of licenses allows us to control what features of the software are to be enabled such as Session Reconstruction. In addition, we can tie the license to a particular machine so that even if the software *and* the license are placed on another machine, the software will refuse to work! Lastly, the licenses can be made to expire so that time limits can be set for how long it can be used. Our methodology is to issue licenses that enable the traffic analysis portion as a rule. When in the course of an investigation of crime, law enforcement needs to do session reconstruction, we will supply the appropriate license with an expiration date matching that of the expiration date of their court ordered wiretap.

NSAT Collector

The NSAT Collector process runs on a dedicated workstation which is connected to the network in a strategic location. The choice of location for placing the collector is the single most important decision in the setup for traffic monitoring. Most sites will want to monitor the traffic that is entering and leaving their network so the location for the collector should be as close to the network feed as possible so that it can *view* the network packets as they travel to their destinations.

The NSAT Collector can be configured using a command line interface which facilitates ease of use from remote locations and batch command files. NSAT support the targeting of up to twenty

addresses in each of the DECnet, LAT, TCP/IP protocol suites. In addition, NSAT can target a packet for collection based on it containing a keyword. These features are discussed below:

- DECnet Addresses: /decnet=(address, address, ...)

The address is in the standard DECnet address format of area.number. An example using DECnet addressing is:

```
nsat monitor/decnet=(16.150,16.143)
```

- LAT Addresses: /lat=(address, address, ...)

The address is the physical address of the LAT device not its LAT service name. Because of this the /lat option can be used to target *any* network devices that you have the physical address of its network interface. The address is input in its hexadecimal notation minus any hyphens. For example the targeting of address 08-00-d2-fc-45-21 would be:

```
nsat monitor/lat=(0800d2fc4521)
```

- TCP/IP Addresses: /tcp=(address, address, ...)

The address is entered in its standard dotted decimal notation. Entire subnets can be selected by substituting 255 for the host portions of the address. For example to target all hosts who live in 16.40.x.x, the wildcarded address would be 16.20.255.255. For example:

```
nsat monitor/tcp=(20.40.5.16,245.15.255.255)
```

In some cases it may be desirable to target only traffic destined outside of the LAN. To accomplish this the NSAT Collector supports intra-area rejection as follows: Given three areas A, B, and C (A and B being local, C being external), traffic between A and B can be rejected whilst collecting only traffic that consists of the external connects (A=>C, B=>C, C=>A, C=>B). Using the /reject option, up to 10 subnets can be selected to have just their external traffic targeted. For example:

```
nsat monitor/reject=(120.60.5.0, 40.1.0.0)
```

- Protocol Number: /protocol=number

Instead of, or in addition to, being able to target packets based on a particular address, the NSAT Collector can collect all packets pertaining to a particular protocol suite. This is initiated by using the hexadecimal protocol number of the protocol you want to target. Up to one hundred different protocols can be selected at one time for collection. To collect all packets that use the Novell protocol the NSAT command would be:

```
nsat monitor/protocol=8137           (Novell)
nsat monitor/protocol=6003           (DECnet)
nsat monitor/protocol=6004           (LAT)
nsat monitor/protocol=0800           (IP)
```

- Keyword Searches /target="string"

The NSAT Collector provides a keyword search tool that allows targeting of packets that contain an occurrence of the string. The string can be any combination of characters up to 32 characters in length.

All of the above options can be used at one time allowing the NSAT Collector to be configured for complex environments. To target the TCP/IP address 35.18.1.121, the DECnet address 25.153, and the LAT device 08-00-de-fc-22-13 you would do the following:

```
nsat monitor/tcp=35.18.1.121
nsat monitor/decnet=25.153
nsat monitor/lat=0800defc2213
```

NSAT Processor

All actions performed after the NSAT Collector has collected the network packets that meet its selection criteria are the responsibility of the NSAT Processor. In an operational environment even the collector itself is controlled by the processor! Components of the processor software are contained on both the collection and processing computers.

On the computer that contains the NSAT Collection software, there is a Batch procedure that runs the NSAT Collector. This procedure is initially responsible for starting the collection process and configuring it according to local specifications. Periodically the Batch procedure will restart and cause the collection process to start dumping the raw packets to a new file. The name of the file is changed to represent the originating system along with the starting and ending date/time combinations contained in the file. The old file is then transferred to the computer containing the NSAT Processor software and the local file deleted. In the case of a failure in the transfer the file is held until the next cycle and the transfer reattempted along with the new file. Because of the size of the raw files it is imperative that transfer errors be corrected within a few cycles otherwise the disk will fill up and collection data lost.

The software on the processor computer takes the raw data files produced by the collector and reduces them into a manageable form called the compressed file. Each connection contained in the raw file consist of many records each representing a single packet. In the compressed file, each connection is represented by a single record containing the start of the connection, type of connection performed, status flags, total data packets transferred, and where possible the total bytes transferred. The format for a TCP/IP connection is shown in Table 1 with corresponding information being used for the other supported protocols, namely DECnet and LAT. It is expected that this approach for collecting information will work well with other network protocols.

The compressed file is then used to update the profile database which contains historical information about which computers are communicating with each other. By querying the profile database, security personnel can determine the type of activity originated or received by a given network address over a period of time. The historical information is presented to the granularity of a day which would then be used to limit the searches of the compressed files for more detailed information on the connection in question.

Two additional databases are used by the NSAT Processor to provide address to name translations. The Address database contains the information to identify the system type, and point-of-contact, date last seen, along with the name of the system corresponding to a given address. There is a record in the Address database for every address that has been seen by NSAT. This is true even if the Address does not have a corresponding entry in the namespace. Corresponding to the Address database is the Networks database which contains information to identify the owners of the network portion of the addresses we have seen. Both databases will support versioning so that the historical perspective on who owned a particular address can be obtained. This is useful when investigating problems that occurred weeks, or months ago and the owners and/or locations of the system has changed.

Field	Type	Description
protocol	byte	Protocol Number
sport	word	Source Port Number
dport	word	Destination Port Number
source	long	Source Address
destination	long	Destination Address
start_time	long	Time Communications Started
sequence	word	Sequence Number
end_time	long	Time Communications Ended
pkt_cnt	long	Number of Packets
sflag	char[6]	Start Flags Seen
eflag	char[6]	End Flags Seen
saddr	char[6]	Source Ethernet Address
daddr	char[6]	Destination Ethernet Address
file seq	long	File Sequence Number

The NSAT report generation capability allows the analyst to review the types of activity being performed from a source address, a destination address, or by a type of connection. Address to name translation is performed by searching the Address database for the address and if found using the name found in the database. When a name is not found in the database the network portion of the address is used to locate the corresponding record in the Network database and the network name used. While using local databases for name lookups *may* lead to a discrepancy with what the current name of the systems is, testing has shown that it is *significantly* faster than querying the namespace and/or Network Information Center for the information. For those users who prefer to do the lookups each time there is an option to the report generator to instruct it to update the Address and Network database at run time rather than periodically as is typically the case.

Information gleaned from the NSAT report can be used to create a rulebase which can be used by administrative personnel for manual detection of hacking events. By examining the patterns of connections and the ports to which they are made, it is possible to detect suspicious activity. Even the time of the connection can play a role in detecting abnormal usage!

Once it has been determined that a network address is attempting to hack into a site, NSAT provides the ability to determine which addresses on the network have been touched by the activity. By using the report generator, a report of all addresses that have had connection with a particular address can be produced. The report will provide an excellent starting point for assessing how effective the attack has been, and which system managers need to be contacted via the POC information in the Address database.

Intelligence information from various sources can be used to flag records in the Address and/or Profile databases. Records can be flagged as being normal, anomalous, suspicious, attempted intrusion, successful intrusion, or mixed depending on which best represents the findings of an incident assessment. Sources of information include: host audit trails (including data gathered with the companion tool - Facility Security Services), network traffic analysis, system managers, CERT, and criminal investigators.

The NSAT Collector provides a string detection capability that can be used to detect unauthorized data transfers or malicious code. Once the appropriate identification string has been worked out for a given piece of code or data, it is possible to detect a network connection over which the targeted information is being moved. The Profile database has a field to keep track of which addresses have been seen transferring targeted information.

Operational Use

NSAT is being used in an operational environment at Wright Patterson Air Force Base to test its effectiveness outside of the controlled development environment. The base houses two Air Force labs, a higher-education training facility, and is the headquarters for a major command. Because of the nature of the work performed by these organizations, their computers tend to be of interest to those outside of the MILNET community.

The base's network is comprised of eight class B networks (e.g. 150.20.*.*) and one class C network (e.g. 150.21.2.*) which contain over 8,000 computers ranging from personal computers to super-computers. The NSAT Collector is placed at the base's connection to the MILNET and is configured to keep track of all connections to off-base computers. This is accomplished by instructing the collector to monitor the TCP/IP protocol suite and reject any connection between the nine networks. This will leave just those connections to computers that are off-base. As a rule all computers connected to the MILNET carry a banner indicating that users are subject to monitoring.

Every three hours the connection information is sent to the NSAT Processor for data reduction and analysis where the traffic patterns are analyzed. The raw file averages about 225MB in size and reduces to a compressed file about 6MB in size. Once compressed, the raw packet file is deleted and the compressed file will have two summary reports generated for review by the security staff.

The first report is called the suspect list and is a report of attempts or successful connections from hosts that we have designated *suspect*. These are hosts that for one reason or another have caused

us some concern in the past. Some of them are known to be distributing pornographic pictures while others are addresses from which attempts have been made on systems located at the base. Most of these addresses have been blocked at the router controlling access between the base and the MILNET but they are monitored anyway to determine if they are still active.

Being a military base there is naturally some concern on connections to base computers from foreign addresses. Although the maximum classification of data on systems connected to the MILNET is Sensitive-but-Unclassified, there is some concern over data aggregation and technology transfer that warrants looking into the information that flows out of the base. While we are not naive enough to believe that attempts to gain information will be limited to foreign addresses, attention is paid to where information is going. To keep track of the foreign connections a report is produced detailing the connections between base computers and those located outside the United States. This is accomplished by examining the hostname associated with each Internet Address to determine if it is a foreign address.

A member of the Base Command, Control, Communications, Computer Systems Security Office (BC4SSO) reviews the reports for any items of interest. Should a connection to a particular computer interest the security personnel, either from a report or via a report from a system manager, historical reports on the connections for a particular computer can be generated. Another investigative technique that is used is that connections to non-standard ports are investigated along with any suspicious connections to standard ports by telnetting to the port and analyzing the characters returned. This will usually allow for the determination of the type of protocol being used and a second connection made using the appropriate tool, i.e. Mosaic.

Whenever an incident is detected, the BC4SSO personnel work closely with the Air Force Office of Special Investigations Computer Crime Investigators (AFOSI/CCI) who are responsible for investigating computer crimes. A good working relationship has developed between the two organizations in that information on incidents flows in both directions. When the OSI gets notification of an incident from one of its sources the BC4SSO office is notified and assists the OSI as necessary. Likewise, when the BC4SSO office detects an incident they notify the OSI and again, assist as necessary. The STEP project team lends technical expertise to both groups to assist in the interpretation of the NSAT findings, to develop new tools to aid in detecting and investigating computer crime and other computer security expertise as needed.

Futures

Work is continuing on the NSAT to enhance its collection and analytical capabilities. During the development and testing of NSAT it was observed that an appreciable amount of our investigative time was spent locating WWW and Gopher sites. These were discovered when we were checking out connections to non-standard TCP/IP ports. When telnetting to these ports, we received the tell-tale return strings indicating that port was offering either WWW or Gopher services. In the next release we would like to enhance the protocol detection capability to indicate whether WWW or Gopher was operating on a specific port in addition to the standard TCP/IP protocols.

Connections to other types of networks other than Ethernet need to be examined. Work currently underway includes enhanced support for FDDI networks. Because of the increase in network

bandwidth (100Mbit/sec versus 10Mbit/sec), additional work must be performed to optimize the code in the collector so packets are not lost. In addition to FDDI, the use of ATM (Asynchronous Transfer Mode) needs to be investigated.

The software contained in the NSAT Processor needs to be expanded to process the DECnet and LAT protocols. Currently the processor only compresses the TCP/IP protocol packets although all three protocols can be collected. Once the other protocols are stored in the compressed database, similar queries to those for TCP/IP can be made to determine what computers are exchanging information and where they are located.

Additional work also needs to be done on the analytical software to relieve the amount of human intervention required to interpret the connection reports. As was mentioned above, NSAT currently produces lists of connections that meet the selection criteria such as address. Reports such as longest connect time, most packets, and most connects along with being able to sort the report by protocol rather than address would make the task of analyzing the output easier. Checks for systematic probing by looking for similar connects to network neighbors would also be a plus! In the long term it may be possible to place an expert system within the NSAT Processor to handle the looking for the run-of-the-mill attacks.

Conclusion

As we have shown in this paper, network traffic analysis, using tools such as NSAT, can be a beneficial aid to improving the security of an organization. By being able to identify the connections on a network and keeping a historical perspective on those connections, security personnel can rapidly detect and respond to security incidents. In many cases, security personnel can be responding to incidents before system managers have even detected them using their host-based detection capabilities such as audit trails!

In the operational testing of the tool at Wright Patterson Air Force Base, one person was able to monitor all traffic going on and off the base to identify suspicious activities. This was accomplished by making use of the automated reports and selective queries against the historical data to ascertain abnormal patterns. With continued work on the analytical portion of the tool it is expected that the amount of work required of this individual would decrease thus freeing them for other duties.

Traffic analysis is not meant to be a replacement for securing individual hosts or protecting networks using firewalls. Rather it is another tool to be used as part of an integration effort to provide a centralized security administration environment to help secure an organization of any size from a continuing threat.

SAGE: Approach to Rapid Development of Trusted Guard Applications

Karen Goertzel, Manager of International Programs
Secure Systems and Services Operation
Wang Federal, Inc.
7900 Westpark Drive — MS-700
McLean, VA 22102-4299 USA

INTRODUCTION

As providers of trusted guard solutions for the US Defense Information Systems Agency, the Naval Research Laboratories, the Federal Bureau of Investigation, the Internal Revenue Service, and several foreign government agencies, developers in Wang Federal's Secure Systems and Services Operation (SSSO) realized that most secure guard applications share the same essential architecture, and a significant amount of common functionality.

With trusted guard requirements appearing in more and more procurements, both in the US and abroad, SSSO developers asked themselves whether, rather than having to design and implement a custom-built application from scratch to satisfy each new guard requirement, they couldn't develop a single standard "guard" framework that would satisfy the vast majority of guard requirements, and which would need only incremental customization to satisfy them.

Having made this "discovery", the SSSO developers undertook to specify the detailed requirements for just such a "generic" (or *standard*) guard framework. At the core of this standard guard framework is a transaction control and execution environment designed specifically to *automate* the enforcement of security policies associated with trusted guard, gateway, and firewall applications. In addition, the SSSO developers determined that many common guard functions could be implemented by standard code that would change very little if at all from one guard ap-

plication to the next. These standard functions include:

- transaction control and management;
- discrete input and output functions for each data path;
- pre-processing of data for each transaction;
- content validation of data for each transaction before transfer through the guard;
- post-processing of data for each transaction;
- configuration of the guard software through a GUI-based tool;
- auditing of guard events and viewing audit logs through a GUI-based tool;
- status monitoring of guard processes through a GUI-based tool.

The customizable elements of the common guard framework are the security policy itself, plus any application-specific mechanisms required to enforce that policy. The guard also supports controlled violations of system security policy but strictly limits such violations to processes isolated in the underlying computing platform. This isolation of security violations ensures that the guard application runs without privileges, easing both the accreditation and the portability of the application.

The guard framework is modular, with discrete functions communicating among themselves via system calls and APIs, enabling the easy integration of third-party software modules to implement capabilities required by the specific guard application. For example, the pre- and post-processing functions of the guard framework could be easily extended to support specific digital signature and encryption algorithms or an EDI message-handling capability. Similarly, the context validation function could be extended to use a third-party natural language processor for parsing text messages, and/or an expert systems engine for defining and enforcing complex security policies.

In addition, the guard framework provides a full set of extensible application programmatic interfaces (APIs) to support services in the underlying Trusted Computing Base or untrusted computing platform, enabling the implementation of various communications protocols, etc., as needed by the specific guard application. By making it easy to link in third party source and/or object modules, the guard framework enables the rapid implementation of sophisticated guard applications with a minimum of custom development required. Thus, a SAGE-based guard will comprise bound units provided by Wang Federal combined with application-specific components provided by the guard developer/integrator.

STANDARD AUTOMATED GUARD ENVIRONMENT

Having defined what a standard guard framework should be capable of, SSSO's developers set about designing the specified framework, which they called the Standard Automated Guard Environment (SAGE™). They determined that the SAGE should be portable between computing platforms — including high-assurance platforms, compartmented-mode workstations, and Class C2-level systems — though Version 1.0 would run on Wang Federal's own XTS-300 Class B3-level trusted computing system.

The XTS-300 was an obvious choice for the first SAGE implementation. It was a trusted computer system with which the SSSO developers were intimately familiar, it had been used and accredited in a number of trusted guard deployments by US and foreign government agencies, and it was the chosen implementation platform for the Defense Information Systems Agency's standard Command and Control Guard (C2G). The XTS-300's B3 security level poised the system to satisfy the vast majority of government requirements, which specify high assurance platforms for secure guard applications. In addition to these considerations, the XTS-300 adds the benefit

of a strictly controlled high-assurance computing environment that rigorously protects the integrity of applications developed to run on it, and greatly simplifies the accreditation of those applications. Like any other application, a SAGE-based guard will take advantage of the integrity protections provided by the underlying TCB.

On the XTS-300, for example, the SAGE guard can exploit the unique high-assurance integrity mechanisms of the B3 STOP™ operating system to protect the integrity of executables and intermediate files used by the guard, and to isolate guard objects (files and processes) based on their Mandatory and Discretionary Access attributes. In addition, the XTS-300 provides strong, highly granular type enforcement, which supports the kind of robust yet flexible Discretionary Access policies required for civilian and commercial guard and firewall implementations. Of course, on other platforms — like all software applications — the SAGE guard can be protected only to the extent the underlying TCB is designed to protect the integrity and enforce MAC and DAC policies.

SSSO developers drew on the client-server computing model and the POSIX concept of a layered computing architecture. SAGE functions are implemented in terms of a Standard Client, for which one exists for each guard transaction, with many transactions running simultaneously within a single guard instantiation. The Client interacts with the Transaction Manager to control the flow of transactions through the guard, while calling the individual Standard Servers to perform each phase of guard transaction processing.

The SAGE layered architecture (Figure 1) is divided into a platform-specific section and a platform-independent (i.e., application-spe-

cific) section. These sections are isolated from one another by "abstraction layers" of APIs. The highest level of abstraction, the SAGE Application Layer (SAL), comprises standard guard functions plus the security policy to be enforced, and any additional services required to enforce that policy (e.g., "dirty word" scanning, digital signature mechanism). The SAL calls the Platform Abstraction Layer (PAL) Programmatic Interface (PPI), a set of APIs that map application-level functions to the underlying PAL. The PAL then provides a second set of APIs that map the standard platform interfaces to the actual implementations of services provided by the underlying computing platform.

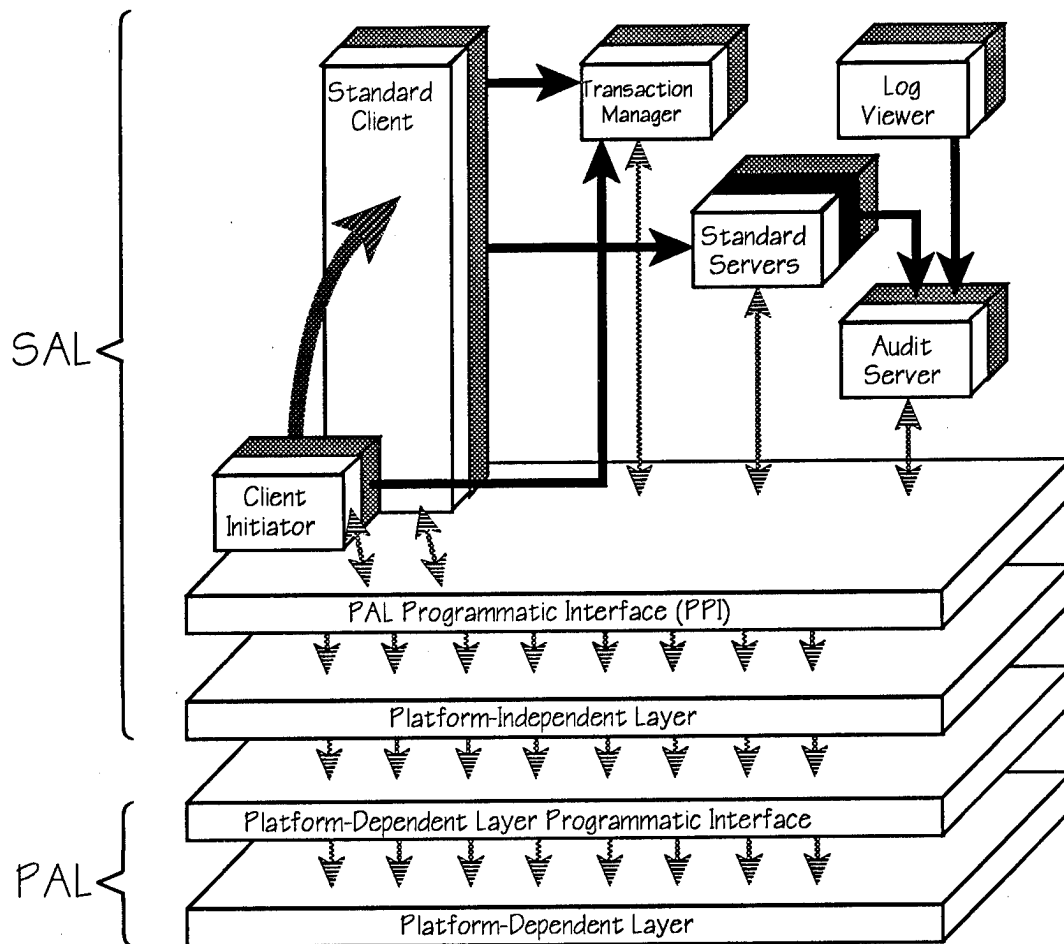


Figure 1. SAGE Architecture

The SAGE abstraction layers contribute greatly to the portability of SAGE and SAGE-developed guards. While platform specifics may vary from computer to computer, the SAGE application layer APIs and abstract interfaces remain isolated from such platform-specific considerations. From their point of view, the underlying system services remain constant from platform to platform. Put another way, from the SAL's point of view, the PAL represents an unchanging set of standard operating system, communications, and security services that a SAGE-based guard can rely on regardless of how the underlying platform implements these services. Figure 1 depicts the architecture of the SAGE, and indicates which of the SAGE modules allow for application-specific extensions/customization.

The SAL

The SAL provides the SAGE-based guard with its platform-independent application functionality. The SAL is implemented as totally unprivileged ANSI C code that is fully portable across multiple platforms. The lack of privileges required by the SAL code not only simplify accreditation of SAGE-based applications, it contributes to the SAGE guard's portability because the guard will not have to be reprogrammed at the application layer to violate the security policy of the underlying trusted computing base or untrusted platform. Instead, SAGE is designed to isolate privileged functions to the platform itself, and provides APIs that enable the guard application to take advantage of the underlying privileged functions to perform security functions such as controlled policy violations, i.e., by making a PPI call to a facility in the underlying platform that is already privileged to perform such violations. In addition, in the ATG implementation (wherein the XTS-300 Monitor is privileged to perform such violations), SAGE ensures that when such violations are requested by the ATG, necessary flow checks are performed, and the violation is logged and audited (if security policy violation events have been configured for audit).

The SAL Standard Client and Standard Servers communicate indirectly amongst each other by passing messages through the PAL. Because it is unprivileged, the SAL cannot violate the Discretionary or Mandatory Access policies of the underlying host system. Any security policy violations required by SAL guard processing are implemented via APIs from the SAL to the PAL, and from the PAL to a facility in the underlying platform that is privileged to perform such violations. In the XTS-300 implementation of SAGE, this facility is the Monitor.

The SAL comprises the following software modules, all of which are delivered with the Application Template Guard (ATG) in SAGE Version 1.0.

- *Client Initiator*—The Client Initiator determines when a new transaction is ready to be processed, which causes a Standard Client to be initiated to process that transaction. For each guard application, the Client Initiator is customized to include an application-specific mechanism to “trigger” its determination that a transaction is ready to be processed by polling a directory on the source system (i.e., the host from which data will be transferred into the guard).
- *Standard Client*—The Standard Client controls the flow of a particular transaction through the guard, and sequences the transaction through the Standard Servers listed in the SAGE configuration file.
- *Standard Servers*—These stateless daemons are called by the Standard Client, and in turn call various SAGE Common Support Routines, PAL PPI routines, and application-specific routines to perform their specific functions. These servers include the Input, Pre-processing, Content Validation, Post Processing, Output, and Termination Servers. Each Standard Server is modular and fully extensible, allowing developers to easily “plug in” application-unique functions. For example, the Content

Validation Server may be extended to call a standard COTS natural language processor to parse complex messages, thus enabling the Server to determine whether those messages meet or violate the defined security policy being enforced.

Similarly, the Pre- and Post-Processing Servers may be extended to call a government-furnished encryption mechanism to decrypt and re-encrypt data after they are processed by the Content Validation Server. Though the actual communication mechanism(s) used to transport data to and from the SAGE is implemented in the Platform Abstraction Layer (PAL), one of the functions of the Input and Output Servers is to call on the PAL to provide the service required to transfer data into and out of the guard environment.

- *Log Manager*—The Log Manager provides a common audit collection service to the Standard Servers and the Standard Client. The SAGE Log Manager can be configured to collect a security-administrator-defined subset of SAGE log events and pass those events to the SAGE log file; the Log manager may also be programmed to pass the events to the PAL for writing to the system audit file. In the latter case, the application-level guard events are combined with system-level audit events to provide full auditing of all guard-related, security-relevant activities. SAGE also provides a Log Viewer, which enables the administrator to see a detailed display of guard log records.
- *Transaction Manager*— The Transaction Manager is the SAGE server responsible for maintaining transaction state information and the inter-process data space used by the customized application components (ie, the Standard Servers). The Transaction Manager supports the stateless operation of the Standard Servers by providing transaction identification, and by monitoring and managing the state of every transaction as it is processed through the guard.
- *X-Windows-based ConfigurationTools and Guard Status Monitor*— The SAGE provides an X-Windows-based Status Monitor for observing guard functions in process, as well as a set of X-Windows based system configuration tools, eg, for defining the list of Standard Servers to be called by the Standard Client and the guard events to be logged by the Log Manager.

The PAL

The SAGE PAL comprises:

- *Platform-Dependent Layer Programmatic Interface*—library of software functions providing a standard programmatic interface to platform-specific implementations of system services (i.e., operating system, communications, and security services). The peculiarities of different platforms are hidden beneath this Programmatic Interface, thus ensuring the platform independence of SAGE-based guard applications.
- *Platform-Dependent Layer*—performs the actual low-level transaction routing and process management. In the XTS-300 implementation of SAGE, these functions are performed via privileged processes (i.e., daemons). The PDL is a library of platform-dependent interface routines used by both PAL and SAL processes. It includes the *Monitor*, which is responsible for guard start-up, shut-down, and for routing and relaying of messages between the Standard Clients and the Standard Servers after authenticating and validating those Clients and Servers. The Monitor ensures that a Server communicates only with a Client at the same classification level unless an explicit re-grade is requested of the *Regrader daemon*. The *Regrader* receives messages from the enforcement modules of the Standard Servers requesting it to perform any violations of TCB security policy required to enforce guard security policy (e.g., downgrading or upgrading). Only the *Regrader* is privileged to perform TCB policy violations.

TRANSACTION FLOW THROUGH A SAGE-BASED GUARD

The SAGE guard model is based on a client-server model with platform-specific communication mediation. All communication takes place via messages sent between the PAL and the Standard Client and Standard Servers. No direct communication occurs between the Client and the Servers.

A transaction is processed through a SAGE guard in a number of steps that are scheduled and managed by the SAGE Standard Client, which interacts with the SAGE Transaction Manager to track of the progress of the Standard Client's particular transaction. The SAGE Transaction Manager acts as a database server of sorts, storing all transactions' ID tokens, as well as other information about the transactions (including the current security level of each transaction, i.e., its Mandatory or Discretionary Access level), plus buffer spaces of fixed size to be used as work spaces or "scratch pads" by the transactions.

Step One: Transaction Initialization

The guard "idles" in a steady-state condition until a trigger event occurs. The Client Initiator recognizes this event and requests a transaction-id from the Transaction Manager via the PAL-level Monitor. The Client Initiator then requests the launch of a Standard Client process, passing it the transaction-id. The mechanism the Client Initiator uses to recognize the trigger event and depends on the requirements of the specific guard application. For example, the event might be a call from an external process, or it might be the mounting by the SAGE guard of two NFS file systems, one on a remote "low side" host and one on a remote "high side" host. In any case, the logic for Client Initiator trigger recognition is implemented in the application-specific portion of the Client Initiator.

From this point, the Standard Client takes control of the flow of Server invocations required to process the transaction. The Servers to be

invoked are listed in the SAGE configuration file. If any Server returns a failure, the Standard Client logs the failure and calls the Termination Server to unregister the transaction and perform any application-specific housekeeping. If all Servers succeed, the Standard Client logs the success, passes the transaction out of the guard to the destination point, and calls the Termination Server to unregister the transaction and perform any application-specific housekeeping.

Step Two: Input Processing

Input processing is handled by the Input Server. Input processing brings the data into the Guard for processing, ie, security policy enforcement processing. In the one-way Application Template Guard delivered with SAGE Version 1.0, the Input Server ensures that data transferred from the originating system are stored in a high-integrity interim directory on the XTS-300. While they reside in this directory, the data are available to the processes that implement subsequent guard processing steps when those processes call application-specific logic. How data are handled as they transit the guard in other SAGE implementations is an application-specific detail.

Step Three: Pre-processing

Pre-processing provides the logic for such application-specific functions as transaction format processing (i.e., processing that confirms that the guard is working with well-formed data records). Pre-processing can be customized to perform functions such as:

- parse packet headers to validate header structure;
- extract identification information from packet headers;
- compute digital signatures or recompute checksums;

- extract identifiers, sensitivity labels, date/time stamps, etc, to be maintained by the Transaction Manager to ensure their correct reapplication to the data after content validation;
- generate messages caused by failed pre-processing (to be forwarded to the Standard Client);
- audit security events according to defined audit collection criteria.

Step Four: Content Validation

The Content Validation Server will be the heart of most more complex guard implementations. This process can be customized to validate transaction content against the application-specific security policy. Content validation provides the logic for application-specific functions, such as checking of data content against application-specific security policy rules, for calling the TCB-level function(s) permitted to violate system security policy, i.e., to enable downgrading, and for enforcement of guard policy rules in onward processing of the data by subsequent guard standard servers.

Step Five: Post-processing

Post-processing provides the mechanism for outbound transaction format processing; it applies any necessary data formatting expected by the target system. The Post-processing Server contains the standard template that can be customized to construct the logic for such application-specific functions as affixing a

digital signature and/or encrypting the outbound data. In the ATG, this server re-grades the file from low to high.

Step Six: Output Processing

Output processing provides the mechanism for transferring the data from within the guard to the destination location. In the ATG implementation, this data transfer occurs from the SAGE host directory to the target directory on the destination system.

Step Seven: Termination Processing

Termination processing standard logic provides a template for constructing the application-specific logic for terminating processing for each transaction. For example, termination processing might be customized to perform the following functions:

- retrieve from the Transaction Manager the identifier, sensitivity label, time/date stamp, etc, extracted by Pre-processing; reapply to the data;
- delete transaction-related data (e.g., data being transferred through the guard, scratch-pad data) from the SAGE host directory upon success or failure of the transaction;
- generate acknowledgement/non-acknowledgement message for forwarding back to originating system;
- audit security events according to user-defined audit collection criteria.

SAGE DELIVERABLES AND ACCREDITATION PHILOSOPHY

In implementing their SAGE design, SSSO never lost sight of their original intent: to create a transaction processing environment and toolkit of functions and APIs from which guard applications could be easily assembled and customized. As a result, the first release of SAGE includes just these

components: a transaction execution and control environment based on the client-server model, a series of C-language libraries of basic guard functions and interfaces, and a library of C-language APIs to lower-layer (PAL) services (in Release 1.0 these are XTS-300 services). As noted, Release

1.0 of the SAGE Application Development Environment runs on the XTS-300 running STOP 4.0.3 or later as well as the STOP Software Development Environment.

As delivered, the Release 1.0 SAGE Application Development Environment includes:

- executables for PAL routines, compiled and linked for the XTS-300:
 - command to manage guard execution
 - XTS-300-dependent Monitor, RCE, and Regrader functions
 - other daemons required to support platform-dependent functions
- executables for SAL routines, compiled and linked for the XTS-300:
 - basic routines
 - Client Initiator
 - Standard Server routines
 - Log Manager and X-Windows-based Log Viewer
 - Transaction Manager and X-Windows-based Status Monitor
 - Standard Client;
- executables for the administration utilities;
- object code for the customizable SAL and PAL configuration files and X-Windows-based configuration tools;
- object code libraries for non-application-specific portions of the Client Initiator and Standard Server routines;
- source code templates and makefiles for application-specific portions of the Standard Servers and Client Initiator routines;

- SAGE Application Layer (SAL) header file and function library;
- PAL Programmatic Interface (PPI) header file and function library;
- tools to compile the application-specific components of the Client Initiator and Standard Servers, and to link these components with the non-application-specific binaries to create the customized application-specific Client Initiator and Standard Servers;
- tools for building the SAGE-derived guard installation media.

SAGE Version 1.0 on Wang Federal's XTS-300 uses the inherent process isolation properties of the TCB to protect the integrity of SAGE guard executables. SAGE isolates its privileged code to a very limited, controlled subset within the Platform Abstraction Layer, dramatically reducing the amount of security-aware application code, and easing accreditation of SAGE-based guard applications.

Having weathered their share of system security accreditations, the SSSO developers knew they had to write SAGE in platform-independent ANSI C according to strict coding standards, and follow DOD-STD-2167A development methods in all accreditation documentation, to ensure consistent, portable, easy-to-accredit code.

The SAGE Development Environment includes a generic set of user documentation (ie, Software Release Bulletin, User's Guide, and Security Administrator's Guide). This documentation is delivered in hard copy and on electronic media, and is designed to be customized by the guard developer to reflect application-specific implementation details. The SAGE Development Environment also comes with a SAGE Application Programmer's Reference Manual. As with the SAGE software itself, all SAGE documentation is maintained by SSSO under strict configuration management control in anticipation of accreditation

requirements. In addition to the generic SAGE system documentation, Wang Federal will make any proprietary SAGE de-

velopment documentation available to qualified accrediting agencies as required.

USING SAGE

SAGE provides the programmer with a well-structured framework within which he can build trusted guard applications more quickly and easily than were he to develop those applications "from scratch". The Standard Server Framework provides not only the entry and exit points to and from each of the standard servers, it provides the transaction management logic for handling of each transaction from initialization to termination. The Standard Server Framework (Figure 2) simplifies the definition of the security policy. Depending on the policy, the programmer will determine which standard servers should be invoked, and what each standard server should do. In many cases, defining what a standard server should do will require the integration of custom-built or third-party source or object modules with, or calls to system services from, the standard servers.

For example, a guard with a security policy that included receiving and decrypting PGP

encrypted files, then re-encrypting those files before forwarding them out of the guard, could be rapidly implemented in one of two ways: (1) by compiling the PGP source code with the SAGE Pre-processing and Post-processing servers, or (2) by writing calls from the Pre-processing and Post-processing servers to a library containing existing object code for PGP. In the latter case, each call would include arguments providing the encryption key or seed and the address of space to which the resulting data should be returned for onward handling by the standard server. At compile time, these PGP libraries would be linked into the SAGE, and become part of the resulting guard executable. In this way, numerous existing software functions, such as (but not limited to) packet header parsing, file content format checking, "dirty word" scanning, virus checking, etc., can be easily linked into the appropriate standard server module of the SAGE.

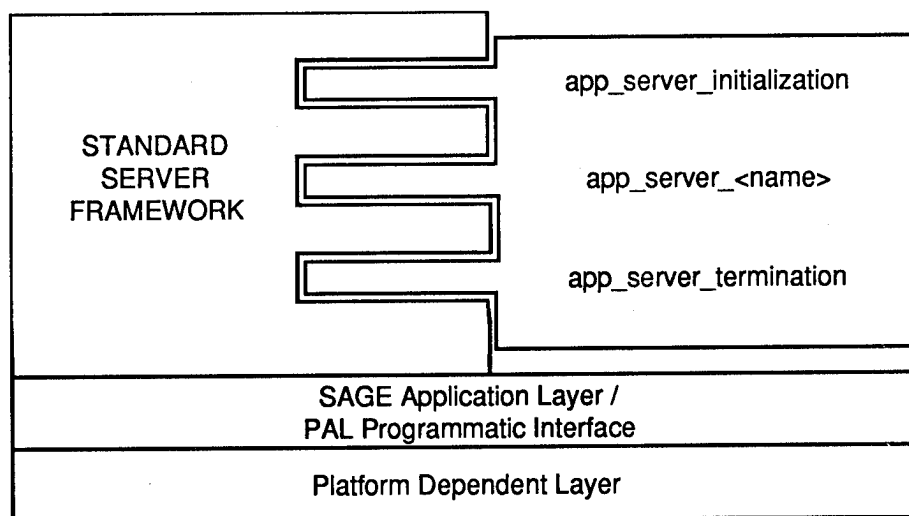


Figure 2. Standard Server Architecture

Similarly, were guard policy to require the use of a system-level service, such as the interface to a Fortezza device, the Pre-processing and Post-processing servers could make the necessary system calls to that system-level interface, including arguments defining the address spaces to which the results of the decryption and encryption requests should be returned.

SAGE's support servers (e.g., Audit server) may also be integrated in this way with third-party software, either through this kind of call-level interface and runtime linking, or through data-level integration. This is already done, to some extent, in SAGE's log files may be "fed"

into the system audit collection mechanism of the underlying operating system (in the case of SAGE Version 1.0, the operating system is STOP on the XTS-300), thus providing a single, integrated audit trail for application and system-level guard-related events. However, this integration of system and application audit files need not end here. The SAGE log file could be further defined to a third-party audit analysis tool such as Haystack Labs' Stalker, and SAGE's log collection mechanism programmed to automatically copy the SAGE guard log files into the Stalker audit database for analysis and reporting.

SAGE USERS

While SSSO originally developed SAGE to simplify their own guard application development efforts, it soon became apparent that SAGE, because of its easy configurability, customizability, and portability, would be a useful tool for other application development shops. Thus, SSSO set about defining the intended SAGE user, and determined that three types of users can take advantage of SAGE:

- *Integration or MIS staffs* tasked to implement a trusted guard application for an end user—Using the SAGE Application Development Environment, Guard, developers are relieved of the necessity of mastering low-level, platform-specific details, and can instead concentrate on high-level issues of guard policy implementation, writing unprivileged, application-specific routines, and binding those routines with the appropriate standard SAGE modules. Thus, the SAGE development environment makes it possible to produce high-quality trusted guard applications based on a uniform transaction model. The result of using SAGE is a guard development effort that is much more cost-effective than building a custom guard "from scratch".
- *End user organizations*, which can use the features of the XTS-300-based "Application Template Guard" — one of the components of the SAGE deliverable — with only minimal configuration—The ATG is a simple, SAGE-derived, low-to-high, non-accredited guard constructed for execution on the XTS-300. The ATG provides the electronic equivalent of an "air gap" between two systems. Using the ATG with the XTS-300 system documentation, an end-user can configure a working guard on an XTS-300 without any additional programming.
- *Integrator and value-added reseller organization marketing personnel*, who can use the ATG as a demonstration tool—Since the ATG is a complete functioning guard with all support tools in place, it can be used immediately for demonstration purposes. The *XTS-300 SAGE ATG Manual* provides user-level documentation of an acceptance test suite that can be used to demonstrate the performance and features of a simple one-way SAGE-derived guard "out of the box".

Experiences with implementing messaging security in MSMail 3.2

Abstract: Our experience with incorporating the MSP (Message Security Protocol) library with the MSMail 3.2 User Agent provides a baseline to judge the complexity of incorporating Messaging Security into other User Agents. MSMail 3.2 is a “layered” and “extensible” User Agent making it particularly adaptable and hence suitable for adding security services. We trade-off the different approaches to adding messaging security to MSMail, leading to the design approach adopted for SPEX/Mail. We relate those elements of infrastructure that are required above and beyond the existing Mail delivery system in order to support secure messaging. An extensible directory service is needed for certificate distribution. Because the proprietary directory systems provided with many mail packages may not be extensible, we have chosen to create our own “adjunct” to the current “address book” functionality of MSMail in the form of the “autograph book”. Infrastructure is also needed to provide for the distribution of Key and Certificate Revocation Lists. We suggest the use of a Web Page on the NSA Web server as a focal point for accessing the latest KRL and CRL files. We review our experiences with performance and interoperability. We review how applicable what we have learned about adding messaging security to MSMail might be to other mail packages, and even other completely different applications. We conclude by reviewing our premise that the experience of implementing messaging security for MSMail has left us with an appreciation for how security may be added to other applications.

Author(s): James E. Zmuda
Russell Housley

Organizational Affiliation: Spyrus

Phone Number: (408) 432-8180

Internet Address: jzmuda@spyrus.com

Point of Contact: James E. Zmuda

Title: Experiences with implementing messaging security in MSMail 3.2

ABSTRACT

Our experience with incorporating the MSP (Message Security Protocol) library with the MSMail 3.2 User Agent provides a baseline to judge the complexity of incorporating Messaging Security into other User Agents. MSMail 3.2 is a "layered" and "extensible" User Agent making it particularly adaptable and hence suitable for adding security services. We trade-off the different approaches to adding messaging security to MSMail, leading to the design approach adopted for SPEX/Mail. We relate those elements of infrastructure that are required above and beyond the existing Mail delivery system in order to support secure messaging. An extensible directory service is needed for certificate distribution. Because the proprietary directory systems provided with many mail packages may not be extensible, we have chosen to create our own "adjunct" to the current "address book" functionality of MSMail in the form of the "autograph book". Infrastructure is also needed to provide for the distribution of Key and Certificate Revocation Lists. We suggest the use of a Web Page on the NSA Web server as a focal point for accessing the latest KRL and CRL files. We review our experiences with performance and interoperability. We review how applicable what we have learned about adding messaging security to MSMail might be to other mail packages, and even other completely different applications. We conclude by reviewing our premise that the experience of implementing messaging security for MSMail has left us with an appreciation for how security may be added to other applications.

Introduction

It's been said that E-mail is the "ethernet of the 90's", i.e. that E-mail is the "lingua franca", or enabling network technology that will allow the Internet to achieve it's real potential of linking everyone to everyone else. I personally think E-mail is a great boon, and routinely use it to solicit technical advice from the "netmind". I have even trusted it enough to order goods and services through E-mail messages where I routinely include my credit card number.

Well, I've had the misfortune of having that credit card number compromised.

In recent memory there have been a number of incidents that have underlined for the community as a whole that commerce on the Internet is not quite ready for prime-time. E.G., the alleged pilfering of credit account information by Mr. Mitnick, etc...

It's my position here that messaging security services can be used to protect against a number of threats to the safety of the Internet.

It's my intention further to show how easy (or hard) it was to add messaging security to a popular E-mail User Agent, namely Microsoft's MSMail 3.2 for Windows User Agent.

In section one we will review the architectural alternatives for adding messaging security to the MSMail 3.2 package. We will conclude with what we consider to be the best compromise between security and efficiency. We also examine how to satisfy the goal of a User interface as faithful to the original MSMail user interface as possible.

In the next section we discuss those elements of infrastructure that are particularly important to secure messaging, but are not necessarily part of the support structure provided by many E-mail packages,

including MSMail 3.2. The most important of these is an extensible directory service that could be used to store and provide for the distribution of MOSAIC certificates. We see that in the near-term before an extensible directory service is available, we will have to provide a directory, or address book adjunct which provides for the storage of certificates. We also discuss how to provide for the distribution of revocation lists.

We talk about experiences with performance and interoperability. In particular, we review the happy coincidence that the Fortezza MSP mail format includes a structured message format from the start, meaning that Fortezza-compliant mailers will provide more than security, that they may also provide interoperability as well. (This is unheard of...a security service actually increasing interoperability!)

We then review how applicable what we have learned about adding messaging security to MSMail might be to other mail packages, and even other completely different applications that could benefit from the security services provided by the off-line, or store-and-forward security model of MSP.

We conclude by reviewing our premise that the experience of implementing messaging security for MSMail has left us with an appreciation for how security may be added to other applications in the most convenient fashion.

But first, we start with a word about why MSP at all. Why not, for example PEM?

Why MSP?

Today, there are three popular approaches to securing electronic mail: 1988 CCITT X.411 Recommendation, RFC 1421, and Message Security Protocol (MSP). Each approach offers similar security services, but there are significant differences. This section summarizes the analysis that was performed to select one of the electronic mail security approaches for SPEX/mail.

DESCRIPTIONS OF THE THREE APPROACHES

The 1988 CCITT X.411 Recommendation provides message content security for X.400-based messaging systems. As such, the security parameters are carried in the message envelope, and the entire message content, which may contain several body parts, is protected uniformly. This approach permits the protection to be added by the User Agent (UA) or by the Mail Transfer Agent (MTA). However, each MTA in the delivery path must be capable of parsing the envelope which includes the security parameters.

RFC 1421, commonly called Privacy Enhanced Mail or PEM, provides security enhancements for SMTP-based messaging systems. PEM provides security of message contents; the security parameters are placed at the front of the protected content, not in the envelope. The syntax is parallel to the syntax used to encapsulate a message content that is forwarded. The content can be a text message or a MIME message. The content and security parameters are encoded to permit accommodate a 7-bit data path provided by SMTP. This approach requires that the protection be applied by the UA, but places no restrictions on the MTA.

The Message Security Protocol (MSP) provides security enhancements for either X.400-based messaging systems or SMTP-based messaging systems. MSP provides security of message contents; the security parameters are placed at the front of the protected content, not in the envelope. MSP defines a new content type which includes security parameters and which encapsulates any other content. The concept of the signed receipt is unique to MSP. This approach requires that the protection be applied by the UA, but places no restrictions on the MTA.

COMPARISON OF THE APPROACHES

The 1988 X.411 approach is invasive to MTAs. Each MTA in the message transfer system must be able to parse the security parameters included in the message envelope. Neither the PEM approach nor the MSP approach requires any MTA changes since no security parameters are added to the message envelope.

The 1988 X.411 approach has some impact on the UA which must apply the protection or communicate to the MTA which protections should be applied. The PEM approach has about the same impact on the UA; the UA must select which security services are desired. The MSP approach has the most impact on the UA; in addition to selecting which security services are desired, the UA must be able to process signed receipts.

All three approaches are content independent.

Since the 1988 X.411 approach includes the security parameters in the message envelope, MTAs can use this information to provide additional services to the user. For example, when the final MTA delivers the message to the destination UA, it can send a protected message to the originating UA indicating that the message was successfully delivered. Neither the PEM approach nor the MSP approach can offer similar services since the security parameters are purposely hidden from the MTAs.

PEM and MSP both provide confidentiality, data origin authentication, and non-repudiation with proof of origin. In addition, MSP also provides rule-based access control and non-repudiation with proof of delivery. MSP carries a security label for rule-based access control decision support, and the MSP signed receipt provides non-repudiation with proof of delivery.

PEM and MSP provide confidentiality in slightly different ways. Both protocols encrypt the message content using a symmetric cryptographic algorithm and protect the symmetric key for each message recipient. PEM encrypts the message key in the public key of each recipient. On the other hand, MSP uses each recipient's public key to derive a token protection key; then the message key, a message integrity check value, the security label, and other protocol control information are encrypted using the token protection key. The PEM approach simply provides confidentiality, but the MSP approach bundles integrity, data origin authentication, and rule-based access control with the confidentiality mechanism.

SELECTION

An informal survey indicated that the signed receipt capability offered by MSP is very important to message system users. This is the primary reason that SPEX/mail implements MSP. The selection of MSP also permits SPEX/mail to secure communications between Microsoft Mail 3.2 sites interconnected by Internet and the Microsoft SMTP Gateway.

Alternatives for adding Secure Messaging to MSMail

The MSMail 3.2 User Agent is not an elaborate mail program. It represents a trend away from large monolithic feature-rich programs toward more smaller programs to provide the same features. This is why the MSMail 3.2 User Agent is constructed in an extensible fashion. The basic features responsible for this extensibility are the support for new message types and the layering of the MSMail 3.0 User Agent.

LAYERING

The basis for the layering in the MSMail User Agent is the API/SPI interface. The API/SPI (for Application Programming Interface/Service Provider Interface) is the interface at which the three services of the Microsoft Messaging Application Program Interface (or MAPI, actually Simple MAPI in MSMail

3.2), namely the Name Service (or "Address Book"), Message Store, and Message Transport, are provided and consumed.

This layering means that applications can be mail enabled simply by adding calls to Simple MAPI. Thus the Mail package doesn't need to provide for handling all types of uses of E-mail. <>Scheduling, for example, is handled through a separate program, Schedule+.

This layering also means that new and completely different mail transport systems can be supported simply by creating a "provider" that supports the Service Provider Interface. This means that the same Mail-enabled applications, and indeed the MSMail 3.2 User Agent look and feel can be used with completely "foreign" mail systems. CompuServe, for example, supplies a provider that allows MSMail to be used to send and receive CompuServe mail.

Figure 1 illustrates the layering of the MSMail 3.2 system.

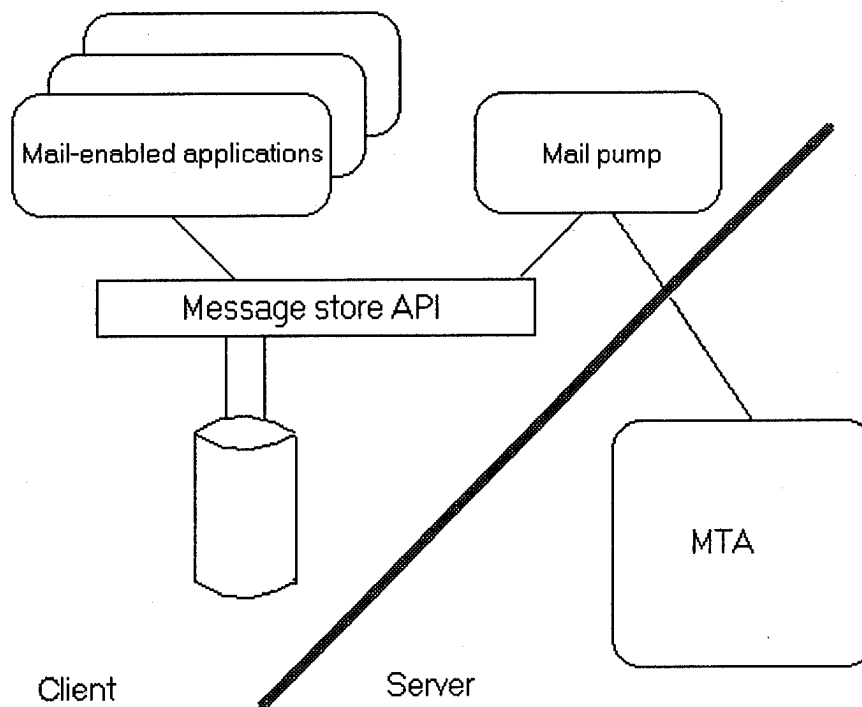


Figure 1 - Layering in MSMail 3.2

ADDING NEW MESSAGE TYPES

The second element that makes the MSMail 3.2 User Agent extensible is the support for extending the types of messages that can be processed. This is known as "adding custom message types". At the occurrence of any event dealing with a mail message, e.g. a Create event, or a Read event, etc...MSMail will determine the program to pass this event to based on the type of the message being operated upon.

This is the established interface for providing for new types of messages. For example, if a MSMail 3.2 site wanted to add its own E-mail meeting notice form, all that would be required is to add a line to the MSMail initialization file, MSMAIL.INI, that lists the newly created message type and the name of the executable file to invoke upon receiving any messaging events for that message type.

This is illustrated in figure 2.

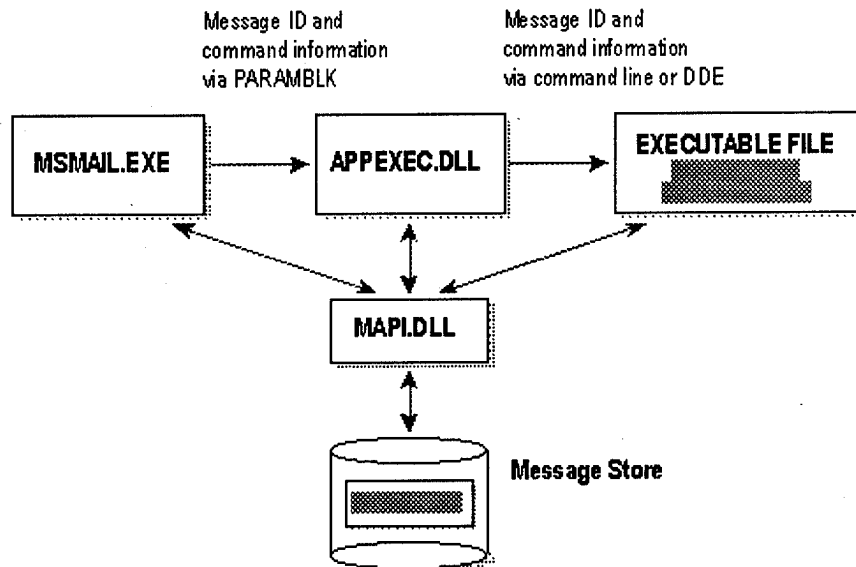


Figure 2 - Adding new message types

ALTERNATIVES FOR ADDING SECURITY SERVICES

Thus it can be seen that there are at least two ways in which to add security services to MSMail 3.2.

First, a new provider could be constructed such that the existing applications would be able to request services through the same MAPI calls that are currently used to provide for the three basic services of Name Service, Message Store, and Message Transport. This provider could then in turn call the underlying MAPI service provider after having manipulated the messages to provide for the security services requested on a transmit operation, or to undo the security services present in the case of a receive operation.

The second way would be to implement the security services in a separate application that is invoked whenever a message of the appropriate type is received. The APIs for doing this are more or less well defined depending on how tightly integrated you want the message to be. Doing loosely integrated message types is relatively easy: use the message store API to create a message with whatever fields you want, use your favorite programming environment to display it, and create an INI file entry to put it on the mail menu. The drawbacks to this approach are that you don't get all the nice behavior of standard mail message viewers (MDI child of Mail main window, sequencing through the folder message list, etc.), and that you can't reuse code from the standard message viewers (double-clicking on recipients to view details and such). Solving those problems requires you to write C++ code and get intimate with a lot more of the mail program. Schedule+ actually uses both approaches: their main program has its own viewers for its own message types, but they've also written a viewer DLL for the mail program that presents schedule messages in a tightly integrated way.

SPEX/MAIL'S METHOD OF ADDING SECURITY SERVICES

In our design we have chosen to use the second method of adding new messaging extensions, namely by adding a new "custom" message type. This is the "approved" method of adding new message types to MSMail 3.2. Figure 3 depicts the architecture of SPEX/Mail as an MSMail 3.2 "custom message type" extension.

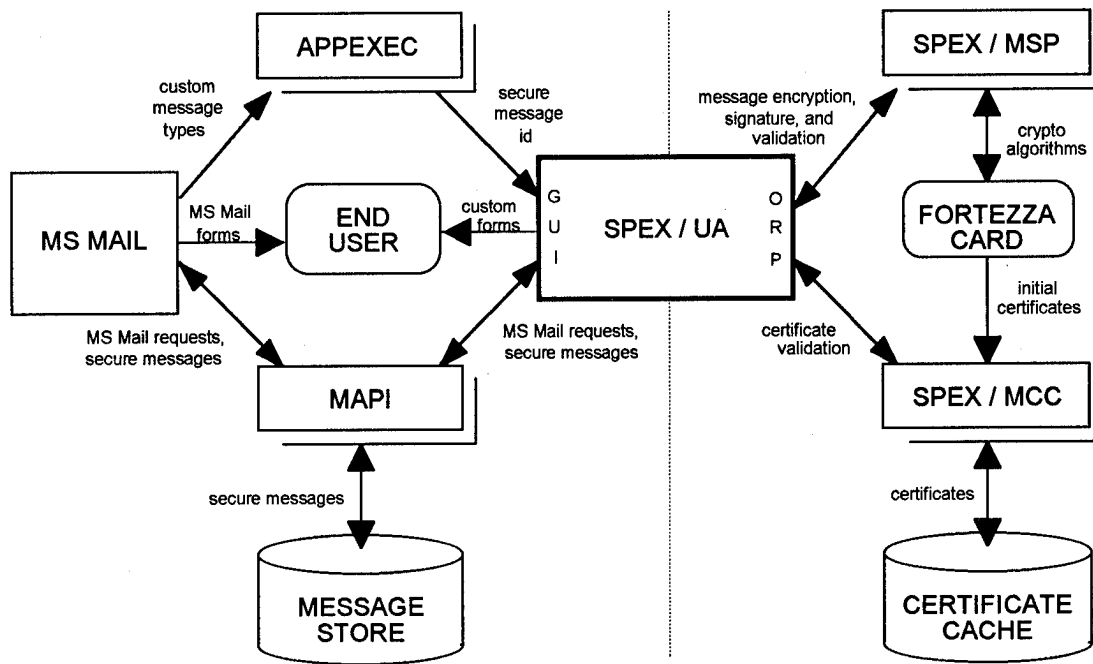


Figure 3 - SPEX/Mail architecture

One of the benefits of this approach are that information is only decrypted while a message is being actively read or processed on the users workstation. Mail messages are always stored, even in the users local folders, in encrypted form. This maximizes security, however, at the expense of some efficiency.

This approach also provides for an independent executable, which can be utilized for handling the display and creation of secure messages for both the E-mail environment, and other applications. As one example, we shall see later that our SPEX/Mail "viewer" may provide the basis for a secure web browser.

Throughout the design our goal has been to maintain fidelity to the original MSMail 3.2 User Interface.

LAYERING WITHIN SPEXMAIL

Layering has been used within the design of SPEX/Mail. This layering accomplishes the same ends that it does in MSMail: it provides the flexibility of swapping out various components. Within SPEX/Mail the main layers are:

- SPEX/msp - This is our MSP 3.0 compliant MSP library
- SPEX/mcc - This is our Mosaic Certificate Cache to provide for the efficient validation and handling of user certificates
- MIME 1.0 compliant user agent layer (with Fortezza MIME extensions) - For encoding MSP messages so they can be processed by "SMTP" mailers.
- SPEX/Mail "Autograph Book™" - Our informal local cache mapping E-mail addresses to user certificates. We see in the next section why this is needed.

With this layering, we have isolated the bulk of SPEX/Mail processing from the details of cryptographic algorithms and certificate formats.

INFRASTRUCTURE TO SUPPORT SECURE MAIL

Secure messaging, even more than traditional messaging, relies upon an up-to-date universal directory service. While a directory service may be helpful when sending an ordinary mail message, e.g. helping the user map a person's name to an appropriate E-mail address, clearly a directory service is not essential to accomplishing this task. All that's really needed to accomplish this mapping is a business card. On the other hand in order to send a user an encrypted message, the originator needs to know the recipient's certificate. Each certificate is approximately 500 bytes. This isn't the sort of thing that will fit on a business card. Clearly some sort of electronic directory service needs to be accessible for secure messaging to work.

Unfortunately, just the sort of directory service that SPEX/Mail needs is not, generally speaking, available in conjunction with proprietary mail packages like Microsoft's. This is because we need an extensible directory that will accommodate a new, very large attribute, namely the certificate.

What this means is that we had to put together an "adjunct" to the MSMail Name Service, which we call the "Autograph Book™".

In the Autograph Book™ we maintain a mapping between E-mail addresses and certificates.

These mappings are established in the normal course of processing received messages. The act of receiving a signed or encrypted message from someone else initiates an update of the Autograph Book™ entry for that person. This occurs only when the new certificate is different than the old, and then the update only takes place if the user confirms the update. Since it is not possible to send an encrypted message without first obtaining a certificate, the initial phase of communicating with a new party is to exchange signed-only messages with each other. This initializes the Autograph Book™ on both sides with all the information necessary for operation.

Figure 4 depicts the "Autograph Book" being used to map E-mail names to Common Names from the certificate.

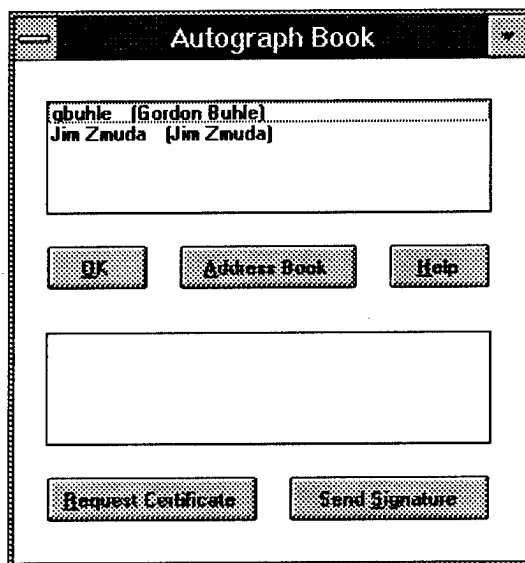


Figure 4 - Autograph Book

The Autograph Book is, of course, only a temporary solution. When an extensible directory service is available, another approach should be taken to providing an E-mail address to certificate mapping.

The lack of an extensible directory service also means that a new way has to be found for distributing the Revocation lists that any public key certificate-based authentication scheme needs. Initially, we see MSMail system administrators obtaining the latest CRL and KRL files and distributing them as file attachments in a signed-only message to those under his administration. We suggest the use of a Web Page on the NSA Web server as a focal point for accessing the latest KRL and CRL files. These can then be forwarded by local system administrators via E-mail. At the User Agent the normal signature verification processing during message reception and running a program locally to verify the structural validity of CRL and KRL updates will provide integrity assurance.

EXPERIENCE WITH PERFORMANCE AND INTEROPERABILITY

Of course one of the first issues that will concern anyone employing security is the impact it will have on the usability and performance of their system. As we've noted previously we have tried to maintain fidelity to the original Microsoft MSMail 3.2 user interface. In the area of performance our experience has been that the initial login process to the card is the most time consuming aspect of utilizing the SPEX/Mail secure messaging system. The actual time overhead of the cryptographic processing, for the most common message sizes (<1024 bytes) is negligible. Actually, due to the store and forward nature of E-mail, all the cryptographic processing times are eclipsed by the time it takes for the mail itself to be delivered.

As far as interoperability issues are concerned, we have actually been impressed with an apparent contradiction that occurred after installing the SPEX/Mail package. Interoperability with other mail packages actually improved. In particular, we have been able to successfully exchange messages with Eudora, one of the most popular public-domain MIME mailers around.

This is because the government, in their wisdom, in the standard for Fortezza E-mail formats[7], included not only a standard for how to carry the binary data that results from a Mosaic encryption, but also specified the format for the message content itself, and insisted that it be a MIME-formatted message. The result is that two Fortezza MIME compliant User Agents of any vintage should be able to exchange secured structured E-mail (i.e. file attachments will work between two different mail packages).

APPLICABILITY TO OTHER SYSTEMS

The next thing to examine is the applicability of our experience to other E-mail packages, and other applications. In particular, we have been examining the effort that would be required to provide messaging security on other mail packages. We have concluded that each system will pose additional challenges. In particular, MSMail seems to be unique in ease of extensibility in the PC environment.

Interestingly, the notion of extending the use of Messaging security to other applications has underlined for us the wisdom of adopting the MIME formats for MSP secured E-mail. In particular, another extensible application that may be augmented with the off-line or store-and-forward security services of MSP is a Web Browser. In particular, since Web browsers support the recognition of MIME content types, the addition of security to a web browser will consist of adding a line to the INI file to invoke a program like SPEX/Mail on reception of a "multipart/X-MSP" content type.

CONCLUSION

We now have an appreciation for the ease of adding one kind of security to non-secure applications. The effort is determined more by the existing architecture of the application being modified than by anything else.

References

- [1] SPEX/MSP API Message Security Protocol Application Program Interface, Document Number D940403, Version 1.03, 24 August 1994.
- [2] MOSAIC Certificate Cache Application Program Interface, Version 0.1, 14 April 1994.
- [3] Secure Data Network System (SDNS) Message Security Protocol (MSP), Specification SDN.701, Revision 3.0, 21 March 1994.
- [4] MSP Appendix MOSAIC Algorithms, 23 March 1994.
- [5] MSP Appendix DMS/MSP/MOSAIC Requirements, 23 March 1994.
- [6] MOSAIC Certificate Labeling Format Specification, Version 1.01, 7 July 1993.
- [7] "MOSAIC Simple Mail Transport Protocol Message Format Specification", dated April 20th, 1994. Prepared by the MOSAIC program, NSA.
- [8] MSMail 3.2 SDK reference, Microsoft Developers Network CD-ROM number 7, Winter 1994.

CAN COMPUTERS AND EPIDEMIOLOGY GET ALONG?

HEALTH PROBLEMS IN COMPUTERS

Guillermo M. Mallén-Fullerton MS
Universidad Iberoamericana/Universidad Nacional Autónoma de México
Cerrada Providencia 43
10200 México D. F.
mallen@servidor.unam.mx

Dr. Florencia Vargas-Vorackova PhD
Instituto Nacional de la Nutrición
Vasco de Quiroga 15
14000 México D. F.
florvar@servidor.unam.mx

Dr. Enrique Daltabuit-Godas PhD
Universidad Nacional Autónoma de México
Dirección General de Servicios de Cómputo Académico
Ciudad Universitaria
México D. F.
enrique@condor.dgsca.unam.mx

Abstract

Computer viruses are producing severe damage to many organizations and individuals. The decisions on which measures should be taken to solve the problem are usually made on a subjective basis instead of the cost-benefit ratio of each different alternative. Epidemiological studies provide the basic information to make similar decisions in health.

In this paper we present the results of a project developed by several Mexican organizations to adapt the epidemiological techniques used in medicine to computer viruses.

First we describe the computer viruses problem from a broad perspective, then, a brief description of the basic epidemiological ideas is presented. Finally we have several considerations to be applied in computer virus epidemiological studies and the experience gained up to this moment in the research project that is under development.

Introduction

From a theoretical viewpoint, the existence of computer viruses was pointed out by von Neumann in the forties in his book Theory and Organization of Complicated Automata, however, from the practical viewpoint, computer viruses are much more recent.

The first lab computer virus was made by Fred Cohen in 1983 as part of a computer security seminar. In that seminar, Dr. Len Adelman coined the term virus we use today.

The term virus is applied to routines or programs designed to spread themselves from one computer to another. We will discuss a computer virus definition later in this paper.

In 1984 a couple of viruses were made in a non-research environment. One was created in Bresscia, Italy, and the other in Pittsburgh, PA, being the first to be found "in the wild." From this moment, users started to have a new problem.

The bulk of the computer viruses exists in small computers, however, there are viruses that work in UNIX systems, like the Internet Worm¹ made by Robert Morris Jr. in 1988, and in the mainframe world, like the Christmas Virus of IBM mainframes.

At the beginning, computer viruses were received by users with fear. Most of the common viruses made in this early period were not particularly damaging, and some produced attractive visual effects, like the ping pong virus of 1987-88. Many users changed their minds, regarding computer viruses as innocent jokes.

There are, of course, many viruses designed as jokes, but many other viruses are specially destructive and there is evidence of an attempt, made by a well known Mexican hacker, and an opposition political party, to introduce a virus in the computers used to count votes in the 1994 presidential election in Mexico, probably to sabotage it. Evidently the extent of the potential damage caused by computer viruses goes far beyond the end user.

Today, we have about 5400 different viruses registered in different catalogs², and it takes a couple of years to double the number of viruses to be found in the wild. There are virtually no users that have not suffered at least one computer virus incident. To solve the problem, a multimillion dollar industry provides different programs designed to detect and remove³ computer viruses from the most popular computers.

It is common to see several similar antiviral programs in the same computer despite their poor combined effectiveness⁴, and the decision on which security measures to use, is frequently made on the price of the product or how well known it is, instead of a serious evaluation of the importance of the problem and the true effectiveness of the product when used in the real world.

There are very few studies to measure the real magnitude of the problem. Many authors rely on the number of different viruses cataloged or the number of virus incidents *reported*⁵, which for many purposes is of relative value, as the total damage caused by a particular virus could be very different from another, some viruses are very common while others are rarely seen in the wild and most encounters with common viruses are not reported.

It is our perception that too much money is spent to solve the problem as a result of ill planned security measures. The real cost-benefit ratio of the different alternatives should guide the decisions, but virtually nobody has a measure of the actual costs and benefits of different technologies and policies operating in the real world.

The first step should be to really know, how big the problem is, what enhances it, and which is the true effectiveness of the most common alternatives available to solve it.

A joint effort between the National University of Mexico and the Iberoamericana University, with the assistance of other institutions, was made to answer these questions. We decided to study the epidemiology of computer viruses in both universities. As a first step, we developed a strict methodology to insure unbiased results and solid conclusions.

To establish a solid ground, we must have a good definition of computer viruses.

Fred Cohen proposes a mathematical definition of computer viruses in his PhD thesis that goes beyond the scope of this work. He also gives an informal version of the mathematical definition in his book "A Short Course on Computer Viruses":

" . . . What makes one of those sequences of symbols [routines] an element of the 'viral set' (V) is that when the machine interprets that sequence of symbols (v), it causes some other element of that viral set (v') to appear somewhere else in the system . . ."

When Cohen's formal definition is used to prove theorems and test some theoretical properties of computer viruses, it works fine, however, in practice it presents some problems. For instance, a disk copy program used to produce a copy of a diskette containing the program itself, would be a virus, while for any practical purpose, it is not.

To solve this problem, Alan Solomon proposes the term "real viruses," but fails to provide a strict definition of it. The idea is to regard a virus as a program or routine designed to spread itself to other programs and computers. For many practical purposes, including our epidemiology study, this is enough.

Epidemiology

The word epidemiology comes from *epi*-about, *demos*-population, and *logos*-study. It is currently recognized as "the study of the distribution and determinants of diseases and injuries in human populations." This definition assumes that human disease and injury does not occur randomly, and that both have causal factors that are susceptible of prevention⁶.

At its beginnings, epidemiology was focused toward the study of infections. Awareness of epidemics of non-infectious problems such as lead intoxication and scurvy expanded the focus of epidemiology to all kind of diseases.

The study of distribution evaluates who, where and when is or gets sick. This kind of information is known as descriptive epidemiology, and provides the framework for hypothesis generation⁷. The study of determinants identifies causes and/or risk factors

of diseases. This kind of evaluation, known as analytic epidemiology, implies hypothesis testing in order to associate an exposure to a given disease.

Knowledge of disease distribution and determinants requires frequency quantification. Two measures of disease frequency are used in epidemiology, namely prevalence and incidence. The former reflects the proportion of diseased in a given population at a given point in time, and the latter reflects the number of new cases of a disease that occur in a given population during a given period. Both prevalence and incidence are related⁶. A very infectious disease can determine a high incidence, but if recovery/death occurs in a short period, its prevalence will tend to be low (e.g. influenza). On the contrary, if a disease is not contagious, but has a longstanding course, its prevalence will tend to be high (e.g. cancer).

When two or more measures of frequency are compared, disease related or causal factors can be identified. In this context, a factor that is associated to a higher incidence of a disease is known as cause or risk factor⁶.

In an epidemiology study, the methods used to identify a virus and the environmental conditions in which it can replicate and act are two good criteria to classify viruses. Therefore, in computer systems, it is important to differentiate boot sector viruses, (which migrate only in diskettes), from program viruses, (which can migrate both in diskettes and through a net), and from multipartite viruses (that belong to both groups). It is also important to distinguish the polymorphic viruses, which are difficult to detect and eliminate and can have an important impact in the prevalence of viruses.

As to the methods used in epidemiology, several strategies that can be used to answer questions⁸. According to the presence or absence of a comparison group, studies can be either comparative or descriptive, respectively. Comparative studies are mandatory for hypothesis testing. Descriptive studies, are used for exploratory purposes and lead to hypothesis generation.

According to the degree of the investigator's manipulation, studies can be classified as observational, quasi-experimental and experimental. In observational studies, the investigator observes the occurrence of events. In quasi-experimental studies, the investigator observes and manipulates the observation scheme. In experimental studies, the investigator observes, and manipulates (controls) both the observation scheme and the intervention, which will affect the occurrence of events.

According to the presence or absence of follow-up, studies can be longitudinal or cross-sectional. Longitudinal studies follow subjects along the time. Cross-sectional studies, limit observations to a given point in time.

A study about the prevalence of virus infection in a given laboratory in a university exemplifies a descriptive observational cross-sectional study. A study about the incidence of virus infection in the same laboratory would exemplify a descriptive quasi-experimental longitudinal study. A study about the incidence of virus infection in two

uninfected laboratories to identify risk factors for infection would be a comparative quasi-experimental longitudinal prospective study. The implementation of measures (such as resident anti-virus programs) that are intended to reduce differentially the incidence of infection would be an example of a comparative experimental longitudinal prospective study.

Methods such as random sampling and allocation, are used to avoid selection bias. Single- and double-blind evaluations are used to minimize evaluation bias. In single-blind studies, receptors do not know the type of intervention administered. In double-blind studies, the type of intervention administered is unknown for both receptors and evaluators. This lack of knowledge is intended to maintain the evaluation process similar in all comparison groups.

Depending on the goals to be achieved, there are many different epidemiological studies that can be made on a population of computers. When little or nothing is known about the population to be studied, a cross-sectional descriptive study to learn the prevalence of the different known viruses is the first step. In some instances, it is also possible to make several studies of this kind at the same time, providing the basis for a comparative study.

A Study on Computer Viruses Prevalence

Since the main goal is to detect known viruses, a good, up to date scanner must be selected. When the number of computers to be tested is large, speed should be an important factor to consider.

As normally the existence of viruses in computers can be regarded as an indication of lack of care, the study must be single blind, that is, the researcher should start the study by surprise and with some plausible pretext, like an investigation of the software versions currently used to estimate the update and training costs. This is specially important when it is not possible to test all computers in a few minutes, as the news of the investigation would spread fast and many users could clean their computers.

A diskette with the required programs should be prepared in advance for each computer to be evaluated. Care must be taken to insure that the diskette is not infected before the study and a clean boot is performed. It should be a bootable diskette containing at least the scanner. In addition is convenient to have a program that checks the OS kernel in RAM to be sure that the machine was booted from the diskette, the scanner output can be redirected to a file, to allow easy extraction of prevalence data. Automated operation is very convenient and helps to avoid mistakes.

Timing of the investigation is important. In many organizations a massive clean of all the computers is performed periodically. The best moment to check the machines is just before the cleaning, yielding a maximum prevalence. With the help of mathematical models of the growth of the number of infected machines is possible to estimate the

average number of infected computers. This figure is important to assess the cost of the problem. The other factor used to calculate this cost is the probability of damage for each particular virus. Often this probability is well documented and has a solid basis as usually it is derived from reverse engineering of each virus.

The characteristics of the population of computers should be documented. It is important to know their configuration, connection to a file server, antiviral programs installed and their frequency of use, version of the OS and other basic software, etc. Also, is critical to have a profile of users and operating procedures.

It is possible to separate several populations of computers, based on differences of their users, antiviral software used, operating procedures, etc. It is possible to make statistical tests of hypotheses to find significative differences in prevalence. Analyzing the profile of populations with different prevalence, ideas can be found to solve the problem. i. e.: in two populations the same brand of antiviral software is used, but in one of them a daily batch scanning is performed while in the other the resident module is used, pointing to the selection of a better strategy.

In the recent years the computer viruses problem grew fast due, on one hand, to the growth in the number of PC's installed, and in the other, to the existence of an increasing hacker community.

The Natas virus is making very important damage since the end of 1993. Commercial scanners did not detect all the variations of this virus until mid 1994. Furthermore, it was practically a local problem, as it was not found frequently in other countries, and there was not a single organized group to study and solve the problem in Mexico. The cost for the country was enormous.

It is well known that universities have a high prevalence of viruses because the same computers are used by many students during each day. The National University of Mexico (UNAM), with approximately 14,000 PC's had recently a very bad time fighting the Natas virus. As a result, a small computer virus research group was created with the cooperation of the Iberoamericana University (UIA).

Soon clearly several research lines were required: one disassembling viruses, another studying the theoretical aspects of computer viruses, and a third one looking for the best ways to solve the problem based on the knowledge developed by the former.

A striking difference on the extent of the problem in different areas of both universities, where in some student labs there were practically no viruses while in others there were plenty, made obvious that epidemiological studies were necessary. To avoid the "reinvention of the wheel" an epidemiologist from the National Nutrition Institute was incorporated to the group. The cooperation of the Electronic and Informatcs Technology Center, who provides maintenance to the UNAMs PC's was also very important.

A pilot study was started in three different student labs in the UNAM and another three in the UIA to test the software developed for the project and look for unexpected problems. From the six labs, only two had viruses, one with all the machines infected with Natas and the other with a zoo. Several adjustments were made to the software to accommodate all the cases: standalone PC's with hard disk, networked PC's with and without hard disk.

During march, three student labs in the UNAM and five in the UIA were studied. The main results are presented in table 1.

As can be seen, there are many labs without viruses at the time of the study, while in others there are moderate number of infected machines and in one, the problem is severe. The first striking result is that the labs with a file server and workstations without a hard disk are clean. This is consistent with the fact that the ten more common viruses in México infect the Master Boot Record of hard disks, which does not exist in this case.

Another issue is the profile of the students using the labs. The UNAM labs are used by senior high school students and have a moderate number of infected machines. The UIA 4 lab is used by computer science students and has a severe virus problem and the UIA 5 lab belongs to the mechanical and electrical engineering school and has no apparent problem. A possible hypothesis to explain these differences is the behavior of the students: senior high students tend to copy many game programs, the mechanical and electrical engineering students use just a few programs and have no need to copy programs or exchange floppy disks with other machines while the computer science students have a heavy program and diskette exchange that goes far beyond the university as many students work in companies and government. A different study must be made to confirm this hypothesis or to find another explanation.

The UIA 4 lab has been sampled several times with similar results. We are planning to install special software to detect the moment in which each computer is infected along with other useful data.

In every case the infection detected was Natas virus. There may be other infections masked by Natas because the scanner we used detects only the last infection in the

Table 1. Infections in different labs.

Lab	Stand Alone PCs	PC with net & HD	PC with net & no HD	TSR Antivirus	Periodic scanning	% infected
UNAM 1	13	11	0	no	yes	4.2
UNAM 2	79	0	0	yes	yes	2.5
UNAM 3	27	0	0	yes	yes	3.7
UIA 1	0	0	60	yes	yes	0
UIA 2	0	0	30	yes	yes	0
UIA 3	0	0	80	yes	yes	0
UIA 4	13	0	0	no	yes	54
UIA 5	15	0	0	yes	yes	0

case of Master Boot Record viruses.

We have several other plans designed to detect new viruses early enough to restrict their spread and minimize damage. It is possible to set up bait computers in the appropriate points, where an infection is more likely. This requires the knowledge of the environmental factors that produce a high incidence of infections that will be the result of other epidemiological studies.

Currently there is no national antivirus policy in México. We expect to convince the government to finance epidemiological studies. We are confident that it is possible to reduce the computer virus problem at a very reasonable cost using some simple measures that may be different from those proposed by computer specialists and antiviral software vendors.

¹ A worm is just a particular case of a virus as it spreads itself like any other virus. It has, however, the particularity that it does not modify any other program.

² Almost every antivirus manufacturer has a virus catalog listing the viruses that can be detected and/or eliminated. There are other catalogs, like the Hamburg University Virus Catalog and the Patricia Hoffman's Virus Catalog. The number of viruses listed in each catalog is usually different, but there is a consensus: we had about 5400 viruses in mid 1995.

³ Some antivirus products claim that they "immunize" programs. Only in the case of a few viruses it is possible to really immunize programs. Usually the word immunization is used by antivirus developers to say that a checksum is calculated for each program in order to detect any change in it. This is just another way to detect a virus and not a real immunization.

⁴ There is a large redundancy in the viruses that are detected and/or removed by the serious products. Very new viruses are not detected by any product. Using more than one product often yields no better protection.

⁵ Some examples are: McAfee, John, **Computer Viruses, Worms, Data Diddlers, Killer Programs, and Other Threats to Your System** St. Martin Press, New York 1989; Tippet, Peter "Is the virus Problem Growing Exponentially?" and Kephart, Jeffrey O. et al "How Prevalent are Computer Viruses?" in the **Proceedings of the Fifth International Computer Virus and Security Conference**, New York, march 1992.

⁶ Hennekens CH, Buring JE. **Epidemiology in Medicine**. Little, Brown & Co. U. S. A. 1987

⁷ Mausner JS, Bahn AK. **Epidemiology. An Introductory Text**. W. B. Saunders Co. U. S. A. 1974

⁸ Hulley SB, Cummings SR. **Designing Clinical Research - An Epidemiologic Approach**. Williams & Wilkins. U. S. A. 1988.

Disaster Recovery Planning Case Study: The South African 1994 Election

Walter Cooke, CISSP

W. J. Cooke & Associates Ltd.
3216 Albert Street
Halifax, NS B3K 3M9
Canada
cooke@uncle.com
<http://fox.nstn.ca/~cooke/>

Abstract: By all measures, the April 1994 South African Election was a historic event. It was dubbed the “mother of all elections” by the South African Ambassador to Canada. The Disaster Recovery Planning (DRP) work for the IT portion of the election is a case study containing both brilliant and bozo decisions, executed under impossible time constraints and the constant threat of civil war. This case study explores the actions of the author who was the consultant responsible for computer security and DRP, the IT team responsible for development of the electoral computer applications, and the IEC staff as a whole. The analysis of their actions may help you prepare robust Disaster Recovery Plans in the future.

The author also describes the election atmosphere, and the experiences which made the DRP work challenging from a personal perspective. Against all odds, the first democratic election in South Africa was deemed free and fair, and a very tired team of workers returned home to ponder this once in a lifetime event, mindful that the outcome could have been very different.

INTRODUCTION

In April of 1994, everything – work, play, love, and hate – moves to a beat in Johannesburg. Hate, of course, has the loudest, most visible rhythm. There is sporadic gunfire from ANC HQ, where the guards get antsy at 2:30AM and empty their clips if they see a suspicious car coming down the street. We use ear plugs to block out this rhythm and find sleep. Bus drivers rhythmically toot their horns while flashing cryptic hand signals to tell potential passengers which township they’re headed for. People walk with an amble that speaks volumes about the new sensation of being able to move freely around Jo’burg. Some white men swagger like cowboys, with sidearms strapped to their hips. In South Africa’s population of 37.5 million, the 28 million blacks have never voted before, and about half of the population is unemployed. The writing is on the wall: “Vote Left and nothing will be right; vote Right and nothing will be left.”

By all measures, the 1994 South African Election was a historic event. It was dubbed the “Mother of all Elections” by the South African Ambassador to Canada. The Independent Electoral Commission (IEC) was set up by the South African transitional government, and given the responsibility for running a free and fair electoral process. The IEC was directed by a team of 17 commissioners, appointed from a number of countries, with a mandate to help facilitate and oversee the electoral process. “South Africa was the largest non-military democratic development mission Canada has undertaken,” reported Ron Gould, Assistant Chief Electoral Officer to Canada, and former appointed Commissioner to the Independent Electoral Commission of South

Africa. But "it was well into January [1994] before the IEC was really in place, with an impossible mandate to carry out an election on April 27th. I arranged to bring in a group of Canadian experts in election readiness planning, in voter education, in training of election officials, in voting by the disabled and prisoners, in public electoral inquiries, in communications, and three election computer specialists. This group worked directly for and with officials of the IEC and, in my view, without them there would have been no South African Election" said Gould.

As a "pigmentationally-challenged" computer security consultant coming from the relatively peaceful dominion of Canada, working for the IEC was both a professionally and a personally challenging experience. My primary task was to build and implement a Disaster Recovery Plan (DRP) for the Information Technology (IT) infrastructure that the election process would depend upon. However, general INFOSEC and physical security practices were also needed to stop a full scale disaster from occurring. Both the technical challenges and the culture shock of working in South Africa often stood in my way. The struggle between white supporters of apartheid and the black majority was a daily threat that appeared ready to explode and rip the country apart. The sometimes violent struggle for power between the major political parties seemed impossible to reconcile. Also, computer security professionals do not build and implement a DRP when they are already in the middle of a disaster. But if you spoke with anyone who worked at the IEC, they would probably tell you the election was a series of daily disasters from beginning to end.

BACKGROUND

Never have so few done so much for so many in so little time. The IT application development group has twenty people. By comparison, the Monitoring Department next door has over 10,000 workers. After an unexpectedly move from the Johannesburg World Trade Centre to a downtown office tower, there are no desks or chairs. The software engineers literally program on their knees for the first week and a half. With phones constantly ringing and people buzzing around, there is an air of barely controlled chaos. Most of the IT people are local consultants. A few of us are specialists from elsewhere. Lunch in the IEC cafeteria is like a United Nations get together, with conversations in a dozen languages and many of the African women beautifully decked out in traditional attire. The colours are a wonderful sight for winter-weary Canadian eyes. Pagers and cellular phones ring to the city's rhythm. People are red-eyed and frazzled. Bomb threats and evacuations are commonplace. The lack of experienced and trained personnel is the most difficult hurdle to overcome. Most people working for the IEC have never voted before, let alone any experience running elections.

We see some of the country's linguistic diversity reflected in the Telkom (South Africa's telecommunications provider) multilingual internet sign-on screen:

```
00000      0 0
  0   00  0 0 0   000   00 00
  0   0 0  0 0 0   0   0  0 0 0
  0   0000 0 00   0   0  0 0 0
  0   0   0 0 0   0   0  0 0 0
  0   00'  0 0 0   000   0 0 0

----- EASY ACCESS
AFRIKAANS : SLEUTEL NUI IN.
ENGLISH   : ENTER NUI FOR SERVICES.
ISIXHOSA  : FAKA UNUI UKUFUMANA
           UNCEDO.
ISIZULU   : NGENISA UNUI UKUTHOLA
           USIZO.
SEPEDI    : TSENYA NUI GO HUMANA
           DITIRELO.
SESOTHO   : KENYA NUI HO FUMANA
           DITSHEBEDISO.
-----
NUI? 901040OZ3U
```

These are only six of South Africa's 14 main language groups, plus 24 sizable "home languages" such as Dutch, French, Tamil, and Portuguese. The electoral education process was a mammoth effort. All of the different ethnic groups throughout the country were told why you might want to participate in a democratic election, what voting means, and how you go about voting. This presented a challenge for some of the "Operation Access" education teams who had to help illiterate tribesmen and women who had never held a pencil in their hands before.

THE INFORMATION TECHNOLOGY

Contrasting this grass roots electoral work, the computer technology used in the election is "rocket science." In less than 10 weeks, South Africa is wired from coast to coast. A high speed TCP/IP network, running over CISCO routers links 41 remote sites. Only one vendor is able to provide an entirely integrated software product suite – Microsoft. This is a critical success factor in getting all of the software components to work together. The network is driven by 25 Windows NT Advanced server-based machines, three of which run multi-processor Pentium-based configurations. The database servers, running Microsoft's SQL Server 4.2.1 provide information to more than 1000 concurrent users. The PCs run Windows for Workgroup 3.11 and Microsoft Office. To support the election process over 400 software modules (screens and reports) are built using Microsoft Access 2.0, Visual Basic 3.0, and REGIS (mapping software), running as client-server applications.

Why would anyone chose a "rocket science" solution using a client-server based architecture for such a mission critical application as an electoral system? The solution was forced on us by a number of constraints: budget, the available computer technology in South Africa, and the lack of large numbers of IT personnel to do the work.

The electoral applications include Personnel Registration and Tracking for 300,000 IEC workers, Geographic Information Systems containing textual and graphic information on provinces, election

districts, and polling stations, Inventory Management Systems, Incident and Event Tracking, and an Adjudication System that contains cases, types, parties involved, and judgments on the 400 courts set up throughout the country to handle legal challenges. The database tables are distributed between the central database and local databases. Network and Database Management Training is provided to the key technology staff scattered through each province. This includes doing secure backups and the emergency restoration of the servers.

Each Windows NT server has a CD-ROM drive and a copy of the latest versions of the OS, network drivers, middleware, applications, master databases, and system documentation, all on one CD-ROM. Rather than fighting with boot disks and backup tapes during a hardware recovery contingency, or a disaster recovery process, the CD-ROM can simply be booted from the server to get up and rolling with the latest basic working system. A new CD-ROM is pressed every few days and sent to the System Administrators for updating their site.

DEVELOPING THE PLAN

Preparing a Disaster Recovery Plan for a project of this size would usually involve a preliminary Threat Risk Analysis (TRA). Our TRA takes about fifteen minutes. It is not a matter of whether a threat is probably, but how long before it happens. Fires, floods, loss of power and communications, bombings, terrorism, assassination of staff, civil war, theft, destruction, and loss of data integrity - just about every threat scenario we can imagine is very likely to happen, or already has several times. The only things we discount are nuclear and biological weapons. We figure the warring parties have lots of low-tech machetes, bombs, and AK-47s but nothing big. Because of the extremely high threat probabilities, it is recommended that additional security work both precede and support the DRP effort.

One of our most curious discussions centers around the protection of sensitive electoral information. The confidentiality of voting returns is normally paramount to ensure complete fairness in a regular election. But with over 300,000 people working for the IEC (many of whom actively support the ANC, NP, or IFP political parties), it is assumed that there will be leaks while the votes are being tallied and before the results are officially announced. So the worst scenario would be to have incorrect information leaked. For example, if someone took bad data and extrapolated a majority win for the far right white Afrikaner party, this leak would start a civil war in a matter of minutes. So it is decided that if we do have leaks, we must leak correct information. In this case, integrity is far more important than confidentiality of data.

Getting safely around Johannesburg requires anecdotal knowledge that is handed on from person to person. From reading the local newspapers, I have a good idea of what parts of the city are dangerous. Several of us, wanting to see more of the city, have maps with circles around the no-go sections, and particular intersections with gangs who kill passing drivers and pedestrians. One curious feature is the daily running of the gauntlet from hotel to IEC HQ which is about six blocks away. Having a security background, I feel that I know the drill: don't travel at the same time each day, don't take the same route every day, and don't attract attention. Wrong. This city has a rhythm that must be obeyed. The regular people on the street are friendly, curious, and helpful to strangers. You go with the crowd, and you go when they go. As it turns out, everyone in Jo'burg goes to work at the same time, using the same streets, and they stick together for protection, like a school of fish. If you go one street east or west of the standard route, you may be in dangerous territory. If you try to move around the streets at other times of day, many streets are deserted, or those you meet are not anyone who you want to hang out with. At three o'clock the stores close, and

everyone heads for home. After hours, you can't help but stick out when you're the only white face on the street. From this I learned that physical security must adapt to the culture that it functions in.

Security sweeps the IEC building with dogs every morning, searching for explosives. Unfortunately, the dogs are also trained to react to the smell of sugar (an ingredient in some types of explosives), and so they also find every candy bar wrapper in the building. "Airport security" measures are used to control everyone entering the headquarters IEC building. Everything coming in is X-rayed. Visitors are politely requested to check their sidearms at the front desk. Access to the basement parking garage is cut off, to prevent a copy-cat New York World Trade Center truck bombing.

Certain political and operational logistic decisions that have already been made dictate what security strategies may or may not be employed. For example, we discuss having armed soldiers at the front door as a terrorist deterrent. However, uniformed soldiers would give the impression that the same old repressive apartheid police state is running this election. Instead, security decides to use plain clothed soldiers inside. I question this decision with the head of physical security. After all, isn't intimidation a useful deterrent for keeping the hostile elements at bay? Who wants to be a sitting duck?

To direct the IEC operations during an emergency, a "War Room" is built inside the IEC HQ building. However, one security officer has already told me that given 2 hours notice the Zulu supporters of the IFP can raise 100,000 protesters to march on Johannesburg. If they have two days to organize, they can besiege the city with 500,000 marchers. I suggest that if the building is over-run by 100,000 angry Zulu protesters, their war room may not be in the world's safest place. The head of the army force protecting the IEC facilities offers to "stop every train and bus coming to the city." This would keep most of the protesters out in the homelands, but would also interdict tens of thousands of innocent black workers trying to commute to their jobs in Johannesburg.

To ensure the continuity of the most critical operational IT systems, a hot backup site with a server and LAN is created at a secret location in the Mid Rand. Telecommunications deploy a satellite backup system to ensure that the Provincial offices can communicate if the IEC nerve center is destroyed or has to be moved. Mobile satellite units are sent to army helicopter bases, to be flown to alternate communications sites if terrorists knock out an IEC office. We now feel more secure with three systems (microwave, satellite, and VHF radio) to backup the regular fiber and copper network. The engineers tell us that building a 93 million Rand network for the election has been like gathering cobwebs to make something of substance.

In less than 8 weeks the Application Development Division constructs a robust information system that provides the user community with a central repository of all election related information. The system is designed around four abstract classes of information objects: Persons, Events, Locations, Things, and thus becomes known as PELTIS. Insiders acknowledging the incredible feat of the software engineers that seems to be nothing short of magic, affectionately dub the system HOCUS PELTIS.

My fellow IT workers see the Disaster Recovery Plan taking shape and nervously ask about their safety. I try to reassure them by saying that everything is fine; it's now only two weeks before the election. If the far right Afrikaner army really wanted to stop the election, they would have started assassinating us by now. For some reason this comment does not reassure anyone. Later, an

officer from the Canadian Consulate arrives, asking that I fill out a personal data form to assist in the evacuation of Canadian workers. This leaves my fellow South African workers wondering what emergency is unfolding, of which they are unaware.

In retrospect, the stress level of your fellow workers must be closely monitored and honestly respected. Off-handed, cynical jokes will backfire and cause more stress, not less. The morale of your fellow workers can be easily crushed with a single, unthinking remark. Similarly, events that can cause fear, uncertainty, and doubt ("FUD factors") should be managed to minimize their impact. FUD factors can be handled by quietly dealing with them off-line, in private, away from the regular workplace. This can be difficult when many groups are working in close proximity; a separate security office or a quiet corner is preferable to being the center of attention. Confining security and DRP work to a secured office is also preferable for another reason: the work you are doing may be of interest to those working against you. What happens if a member of your disaster recovery team is the person who causes the disaster? What if a member of the immediate staff is actively involved in sabotaging the enterprise you are working to protect?

While this scenario may seem remote, experience in this election says otherwise. During the months leading up to the election, two of the IEC directors overseeing procurement of computer equipment and services acted in an incompetent and obstructionist manner. The Director of Application Development, finding herself blocked at every junction, finally took her frustrations to the highest level of management at the IEC. For political reasons, nothing was done to replace the two managers, who continued to stall and avoid making decisions. The Director of Application Development, knowing that tens of thousands of IEC workers were depending on operational IT systems, adopted a strategy of "it is easier to beg forgiveness than to get permission." She cut the managers out of the loop, and managed the IT effort without their direct involvement. After the election concluded, it was discovered that the two obstructionist managers spent their time running phony companies to fraudulently bill the IEC for hundreds of thousands of Rand in hardware, software, and services that were never delivered. It is frightening to think that greed would drive anyone to endanger the safety of an entire country and its process towards democratic freedom. (The two managers are currently pondering this in a South African prison.)

At T-minus 10 days, the Director of Application Development asks everyone to move to 12 hour shifts. This passes without any reaction, possibly because it only formalizes what everyone has been doing for weeks anyway. A collection is taken up for a tombstone and funeral expenses of another IEC Operation Access worker who was assassinated while working in KwaZulu/Natal. People are very tired, and morale is slipping badly. The Director delivers a brilliant pep talk after having a gripe session to let everyone express their fears and help restore their resolve to carry on. We learn that another 250,000 person Inkatha Freedom Party (IFP) march on our headquarters and the ANC's is scheduled for the week before election day. We discuss plans for moving to a backup site if the building is over-run. The building's backup generators are tested.

I learn from Patrick, a waiter at my hotel, that the staff are staying there over night, rather than going home to Soweto. They fear for their lives if they meet the IFP marchers on Monday morning, and they may not be able to enter the city if it is ringed with soldiers and razor wire. After his shift, Patrick tells me stories about life in Jo'burg under apartheid. Western Transvaal bomb number 36 explodes by a community hall at Makokskraal near Ventersdorp. A returning field engineer describes the local graffiti: "Welcome to Tanzania - Any problems, dial AK-47."

Monday at 4 AM there is a ten foot high wall of razor wire strung across the intersection outside the hotel, stopping traffic from approaching ANC HQ two blocks away. Johannesburg pedestrians seldom run, but at sunrise there are hundreds of people running to find passage through the wire barricades and shelter from the anticipated two hundred thousand strong wall of approaching Zulu protesters.

A siege mentality has gripped everyone, but every day we try to set up inclusive activities to get the team spirit up and help everyone cope with the constant stress. When people begin to buckle under the strain, they withdraw into themselves, become depressed, and stop communicating. Some people sleep at their desks rather than drive home at night. Some are afraid to go to the ATM machine to get money, and so they feel that they can't participate in group outings. To counter stress, guilt, fatigue, and help everyone stay well, we keep a stock of goodies, full coffee pots, and a carbohydrate-loaded cafeteria open. At the end of the evening, we try to get everyone to stop and go to the restaurant down the street for a group pizza. We ensure we always have extra money, food, taxis, and people to act as "gophers" for others who can't take breaks away from the office. Some people have worked for over 50 days without a single day off.

On Tuesday, a miracle happens. The IFP agrees to participate in the election! There is a palpable sigh of relief; hope is visible on everyone's face. We're happy because this event reduces the probability of attacks on IEC buildings by 50%. Unfortunately, the Vryheidsfront Afrikaner army are still busy shooting and bombing. Now all that remains to be done is update the constitution, change our mathematical model of the electoral counting process interpreted from the constitution, and add IFP stickers to the 70 million preprinted ballots, all in the next six days!

At four days before the election, our focus shifts to report writing and away from programming. Few of the user areas can articulate what they want to see on election day or which database fields are most important. We play "what if" and imagine what reports they might like. Access 2.0 allows a programmer to complete about 10 good reports a day. A multimedia expert arrives to design the graphics presentation system for displaying the televised election results. I'm in a mad rush to complete the Disaster Recovery Plan so it can be tested.

The most complex application is the Seat Allocation System – a set of programs that take the raw number of votes that are won by each party and convert them to seats in the legislatures. Unlike the Canadian system – a constituency based and "winner take all" system, the South African system is based on proportional representation where the electorate votes for a party list. Each party submits an ordered list of candidates names in advance of the election. Seats are awarded in proportion to the number of votes cast to that party. There are 19 parties running in this election. The actual conversion of votes to seats involves some 34 complex equations to get the final result.

The transitional government has set out the Constitutional rules for counting ballots and rolling up the scores. However, the transitional Constitution can be interpreted in several ways, depending upon your point of view. The Seat Allocation System as it stands will tend to quickly round off and exclude the members of the smaller political parties. Our test results are validated, but the Director asks if this is what they really want to have happen when the votes are tallied. Nelson Mandela asks that the Constitution be interpreted in such a way as to be as inclusive as possible. He wants the roll up process to include any possible minority parties in the final result. Work resumes on a formal mathematical model of the constitution, and twenty-four hours later we have a new working scheme that is inclusive and avoids excluding the smaller parties as the results are

rolled up. This must be one of the few times that an information system has directly influenced the Constitution of a country.

Sunday, Johannesburg is rocked by a car bomb that kills 10 and injures hundreds. The explosion sets off a screaming din of car and store alarms. ANC headquarters is badly damaged. IEC headquarters goes to high alert. A bomb explodes next to a provincial IEC office, but work continues after the cleanup. Apparently this is the sixth attempt to destroy this particular office – the first five bombs failed to explode. However, the men responsible for the car bomb in Johannesburg are found, and they quickly sing like canaries. This leads the army to a barn where fifty of the bomber's colleagues are discovered busily preparing more bombs.

A bomb explodes outside of the hotel where many of the foreign workers are staying, apparently meant to scare us off and bring the election machinery to a halt. Another bomb explodes in the departure lounge of the Johannesburg international airport, killing several people. Election day arrives and passes, as reports on voting statistics and discrepancies are produced. Part-way through the televised counting process, the tallied votes for several of the parties suddenly jumps by several thousand. The Director's phone rings, and a TV reporter asks if it is true that the electoral computer system has been hacked and the votes altered. A frantic investigation reveals that a data entry clerk has either posted some of the entries wrong, or bungled an attempt to alter the results.

By the end of voting, the monitoring database has collected over 44,000 incident reports. These include everything from shootings and polling stations being burned down, to minor fights in the often mile-long lineups to vote. This database was a vital damage control system that allowed incidents to be instantly communicated via e-MAIL to quick response units. These teams then defused critical situations before they got out of hand. The database also allowed managers to allocate security and observer resources to the areas that had the highest rates of serious incidents and voting irregularities. Several days after voting is completed, the ANC have over 60% of the votes. We run our new mathematical model of the constitution that determines the list of candidates for Parliament. It's finally done.

CONCLUSION

Apartheid ended not with a bang, but instead with a cheer of freedom. Against all odds, the first democratic election in South Africa was deemed free and fair, and a very tired team of workers returned home to ponder this historic event, mindful that the outcome could have been very different. It was a once in a lifetime experience, seeing the end of apartheid, but would we do it all again? Highly trained athletes have come to expect victory or defeat with the narrowest of margins. IT's part in this election almost failed, and only succeeded by the narrowest of margins because of the team's strength, courage, and determination to succeed against overwhelming odds. This was our heart-felt victory. And like the athlete, who wouldn't push their limits for those kinds of feelings?

The author would like to acknowledge the assistance of Robyn Kall, an Ottawa-based consultant, who provided background information on the IEC electoral computer systems and IT's role in the South African election. Ms. Kall was the Director of Application Development at the IEC.

VHA's APPROACH TO CONTINGENCY PLAN DEVELOPMENT

Gail Belles, Deputy Director
Medical Information Security Service
National Center for Information Security
VA Medical Center, Building 203B
Martinsburg, WV 25420

The Veterans Health Administration (VHA) is the largest centrally-directed health care system in the United States. VHA handles over 1 million inpatient visits and over 24 million outpatient visits annually. The automated hospital information system used in VHA supports 171 medical centers, 362 outpatient clinics, 129 nursing homes, and 35 domiciliaries. Over the past 12 years VHA has developed a growing dependence on their automated hospital information systems, and the data and information that has accumulated in these national databases.

There are a number of Federal laws, regulations, and directives that address the requirement for protecting Federal information processing resources throughout all departments of the Federal government. Medical Information Security Service (MISS) utilizes these directives to develop policy and guidance to assist all VHA medical care facilities in developing procedures to protect their automated information assets. Assistance provided includes generic policy development, manual and automated risk analysis tools, contingency plan development and testing, and training of users at all levels in the organization. Extensive guidance, procedures, and training modules are continuously being developed and disseminated throughout VHA to assist medical care facilities with their disaster recovery and contingency planning efforts.

The contingency plan development and testing process can be very time-consuming, but if done correctly, can be the key to protecting an organization from major loss of critical operations. The ability to plan for untoward events before they occur has proven invaluable to VHA when dealing with the impacts of several major disasters that included severe damage from hurricanes and earthquakes.

A tutorial has been developed to provide participants with a thorough understanding of the process used by VHA to develop, implement and test contingency plans in the medical care setting. Specific guidance provided in the tutorial will enable participants to develop contingency plans or address specific areas in their plans that may have been overlooked. The session includes a presentation and discussion of major incidents that required activation of contingency plans as well as a discussion of the impacts on businesses that were not prepared to deal with catastrophic events because of the absence of critical operational information. Participants will be exposed to a wide range of issues that need to be considered when developing viable disaster recovery and contingency plans.

Tutorial Outline

1. Real-life disasters and their impacts
2. Contingency planning defined
3. Goals of contingency planning
 - a. Protect human life
 - b. Minimize risk
 - c. Recover critical operations
 - d. Define recovery strategies
4. Defining policy/responsibilities
5. Components of the contingency plan: an overview of the components is provided, followed by a focused discussion of each of these components.
 - a. Disaster avoidance
 - b. Assessing threats and consequences
 - c. Application analysis
 - d. Service/Section data collection forms
 - e. Critical functions work flow
 - f. Personnel inventory
 - g. Service inventory
 - h. Space requirements/inventory
 - i. Office evacuation
 - j. Recovery strategy
6. Testing and maintenance of the plan

FUNCTIONAL SECURITY CRITERIA FOR DISTRIBUTED SYSTEMS

Janet Cugini, Rob Dobry, Virgil Gligor, Terry Mayfield¹

Abstract

The National Security Agency (NSA) with the cooperation of the National Institute of Standards and Technology (NIST) formed a technical group to create security requirements for distributed systems. These include requirements for data confidentiality, data integrity, cryptography, distributed identification and authentication, as well as for access control, auditing, system management, trusted path, and trusted recovery for distributed systems. These requirements are being reviewed for incorporation within the Common Criteria, which is a joint effort of the United States (NSA and NIST), Canada, France, Great Britain, and Germany to come up with a single criteria for security requirements.

Keywords: protection profiles, security targets, assignment, refinement, augmentation, cryptography, key management, distributed system, secure distributed system, realm, trusted channel, security perimeter, interconnection policies, simple subject, compound subject, delegation chain, restricted delegation chain, AND-chained identities, message origin authentication, mutual authentication.

1.0 Background

The Federal Criteria for Information Technology Security [FEDCRIT] was a major initiative of both the NSA and NIST to revise the Trusted Computer System Evaluation Criteria [TCSEC] which is commonly known as the "Orange Book." The Federal Criteria became available for public review in January of 1993. The focus of the Federal Criteria was the same as the Orange Book, that is, to define security requirements for multi-user operating systems. It was decided to limit the applicability of the Federal Criteria to multi-user operating systems because of the extent of change the Federal Criteria incorporated and because we wanted to allow the public to comment on what we had developed thus far. However, most of the reviewers of the Federal Criteria expressed their disappointment with the fact that no aspects of secure networking were included. Therefore, a task group consisting of the authors was formed to specify the requirements for secure distributed systems in order to bring the criteria in line with today's distributed system technology.

In the meantime, an effort was started to harmonize the Federal Criteria, the European's Information Technology Security Evaluation Criteria [ITSEC], and the Canadian's Trusted Computer Product Evaluation Criteria [CTCPEC] into one Common Criteria [COMCRIT], and the sponsoring countries for this Common Criteria agreed that requirements for secure distributed

¹ The author's addresses are:

Janet Cugini: NIST, Gaithersburg MD 20899, cugini@csmes.ncsl.nist.gov
Rob Dobry: NSA, Ft. Meade MD 20755, dobry@dockmaster.ncsl.mil
Virgil Gligor: University of MD, College Park MD, gligor@eng.umd.edu
Terry Mayfield: IDA, Alexandria VA 22311, mayfield@ida.org

systems should be incorporated into this work.

This paper presents an overview of the functional requirements for secure distributed systems that were developed by the authors and by a task force consisting of experts in the field of secure distributed systems. This work has been submitted for incorporation into the Common Criteria. As the Common Criteria is still being written, this paper will not focus on how these new secure distributed system requirements are presented and stated in the Common Criteria. Instead it focuses on the functional requirements for secure distributed systems that were developed without implying any particular Common Criteria commitment to or organization and wording for these distributed requirements. In order to understand the context for our distributed system work, a quick overview of the Common Criteria framework is also presented.

2.0 Common Criteria Framework Overview

The secure distributed system requirements that were developed fits into the overall framework of the Common Criteria. The Common Criteria specifies how security requirements can be incorporated into protection profiles and/or security targets. A *Protection Profile* is a statement of the functional and assurance security criteria that help to counter a set of threats. Protection profiles can be shared by IT product producers, consumers and evaluators, and can be developed by consumers, vendors, or a single organization. Protection profiles can be tailored to meet the needs of a specific environment. These profiles are product independent, i.e., one protection profile can be used for the creation of many products. After protection profiles are created, they can be evaluated by an independent body to ensure technical soundness. *Security targets* can be thought of as an extension of a protection profile in that a security target shows how a particular product meets a specified set of security requirements. Security targets can be created without the prior existence of a protection profile, and in that case all the security requirements that would have been found in the protection profile will have to be stated in the security target along with the instantiation of these requirements for the particular product. The security target is an integral part of a product's evaluation, and becomes the basis for how a product gets evaluated.

Both protection profiles and security targets are made up of elements of functional and assurance components. Components are sets of indivisible security requirements. These components vary based on the scope, granularity, strength, and coverage of the requirements, therefore one can select the specific component he/she needs to counter the specified threats that are listed with the component. Each component also includes a list of the functional dependencies of the component requirements on other components. The protection profile/security target writer must select requirements that are compatible on the basis of these dependencies.

When creating a profile, operations can be performed on the functional and assurance component requirements so that they can be tailored to meet the needs of a particular environment. For the functional components, the assignment and refinement operations are defined. The *assignment* operation allows one to specify values for any parametric requirements or templates, and the *refinement* operation allows one to give additional specifications to or interpretations of requirements. The refinement operation is important since experience with the Orange Book has shown that many requirements are very abstract and do not properly express an organization's needs. For the assurance components, the augmentation operation is defined. *Augmentation* allows for the selection of a component at one assurance level and the addition of a higher level component element to augment the lower level requirements.

The distributed system requirements were written to conform with this basic framework. As with the other components, the various components that deal with distributed systems can be selected as needed to counter the specified threats as long as they are consistent with the stated dependencies. The distributed system requirements can be used for both protection profile and security target specifications, and as with the other requirements can be tailored to meet the needs of a particular environment.

3.0 Secure Distributed Systems

We define a *distributed system* as being a collection of nodes that are connected by communication links to one or more networks that participate in the routing of messages within these networks. This definition for a distributed system differs from the definition of a network in that the existence of autonomous computers is handled transparently by the distributed system, as opposed to the explicit network addressing that is used to control message transfer in and among networks. Also, unlike a network, a distributed operating system can reside throughout the system as opposed to residing on one network server. This allows applications to be shared regardless on which node they reside. A *secure distributed system* consists of a set of trusted hosts and/or realms that are connected via trusted communications channels that are subject to consistent security policies. A secure distributed system contains one or more security perimeters that are subject to interconnection policies, or constraints, placed on one or several of these security perimeters.

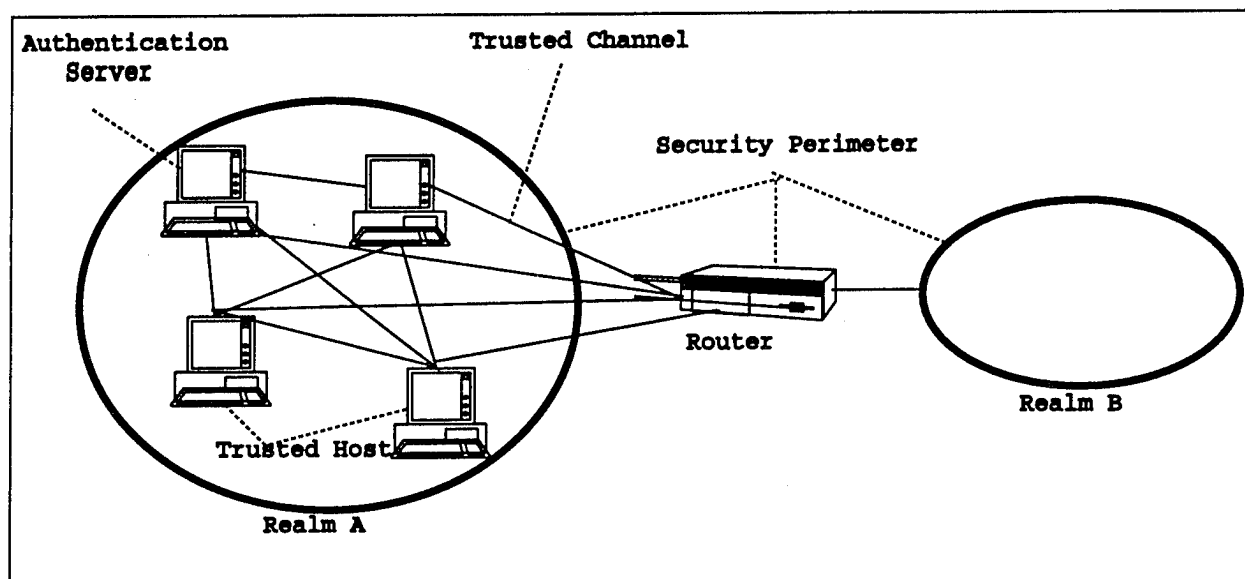


Figure 1. A Secure Distributed System

Figure 1 shows an example of a secure distributed system. As shown in Figure 1, there are two sets of trusted hosts. These sets of trusted hosts are grouped into realms. A *realm* (also known in the literature as a domain or a cell) is the basic unit of operation and administration, i.e., it consists of a group of users, systems, and resources that typically have a common purpose and share common services. The trusted hosts are connected to one another via trusted communication channels. A *trusted channel* is an information transfer path in which the set of all possible senders can be known to the receivers, the set of all possible receivers can be known to the senders, or both. It allows two or more subjects to communicate. A trusted channel for both inter- and intra-realm communication can provide one or more of the following:

- data confidentiality, which enables the sender to know who can read the message it sent,
- data integrity, which enables a receiver to know that the message it received is unmodified and, therefore, also enables the receiver to know who originally created the message,
- authentication, which enables both the sender and receiver to find out who is at the other end of the channel, and/or
- availability, which enables the sender and receiver to use the system at any given time. Thus the sender is ensured that his message will be received.

As indicated by the thick lines in Figure 1, surrounding each realm is a *security perimeter*, which is the interface between the network, the hosts, and the gateways and other realms. A security perimeter represents a partition of a secure distributed system that delimits the scope of administrative control, application resources, and the scope of security policies being enforced by a single, centralized administrative organization. Figure 1 simplifies the connection between the two distinct realms by showing the connection point as being one router (which may perform packet filtering or other types of access control), but it is possible for each realm to have its own gateway and for there to be multiple choke points between any two realms. *Interconnection policies*, or constraints, consist of a set of rules that define whether trusted channels may be established between the trusted hosts within a realm and/or among different realms. It also defines the types of trusted channels (e.g., for confidentiality only, integrity and availability, authentication only, etc.) that can be established subject to these interconnection policies. Figure 1 also shows an authentication server. An authentication server or authority is a trusted agent that acts as a source for certified user/realm identities. A secure distributed system may also contain other trusted servers, such as an audit server, a registration server, a time server, and/or a key escrow authority which would have the responsibility for archiving keys. Also, a realm may contain within it some additional security perimeters and different levels of protection on the trusted channels since not all hosts within the realm may be equally trusted by the other hosts in the realm.

The functional requirements for distributed systems are based on the premise that only the trusted security functions of the hosts and the channels have to be evaluated in order to determine the distributed system's protection characteristics. Therefore, as shown in Figure 1, the requirements for a secure distributed system include the components of centralized-system products (trusted hosts and applications), the trusted channels, security perimeters (which may include routers, gateways, and/or firewalls), and the interconnection constraints.

It is important to note that the secure distributed system requirements we have written do not represent what is commonly known as a security service (i.e., a service which may be invoked directly or indirectly by functions within a system to ensure the adequate security of the system or of data transfers between components of the system or with other systems). There are several reasons for this:

- The requirements that are stated in the criteria must be applicable to the trusted security functions of operating systems as well as for distributed systems. However, most operating system services, including directory, file, inter-process communication, and synchronization services, share similar security functions and requirements (e.g., access control policy, reference mediation). Therefore, per service requirement specifications would lead to significant requirement

- redundancy.
- A service-based requirement specification would inevitably lead to the current "layer wars" in the communication area, since many seemingly similar security requirements appear in several layers of the communication protocols. Controversy as to which service and/or layer is more suitable for a specific security function was avoided by specifying generic requirements that can be used, instantiated, and/or refined in different service and layer contexts as the need arises.
 - Requirement specifications for security functions, rather than for system services, appears to be generally accepted by the security community. This is evidenced by the fact that the Orange Book and its interpretations (the Trusted Network Interpretation [TNI], the Data Base Interpretation [DBI]), as well as the other trusted operating system criteria (Federal Criteria, CTCPEC, and ITSEC), are not created from a security service point of view, and its format is widely accepted.

It is also important to note that the functional requirements for secure distributed systems, as well as for the other functional requirements in the Common Criteria, are application independent. For example, the requirements for centralized or distributed data base systems, secure mail applications, document signature verification, etc., are not explicitly addressed. However, the functional requirements that are in the criteria are intended to be consistent with, and support, these applications without addressing the requirements for these specific applications themselves.

Also, many of these security requirements are written at a level where they must be instantiated (using the refinement operation) to meet the specific protection requirements for a particular organization. In other words, some of the requirements are written as templates. For example, for distributed systems that require cryptography, there is a cryptographic functional requirement that states that the cryptographic algorithm should be selected in accordance with international, national, and organizational standards. This requirement does not state or mandate the use of any particular encryption algorithm. Rather, it is the responsibility of the organization writing the protection profile and/or security target to select the encryption algorithm based on the level of threat it wishes to counter and on international, national, and organizational standards. Likewise, the criteria does not mandate the use of public or private keys or of discretionary or mandatory access control. This gives the protection profile writer a great amount of flexibility in selecting requirements that meet the needs of his/her organization.

This paper focuses on the following functional requirements areas for secure distributed systems: data confidentiality, data integrity, cryptography, and distributed identification and authentication. These areas in particular contain many of the unique requirements for secure distributed systems, and this paper highlights the requirements that were written for these functions. Note that secure distributed system requirements must also be added to other functional areas (access control, auditing, system management, trusted path, trusted recovery, etc.) and to the various levels of assurance.

3.1 Data Confidentiality

The goal of data confidentiality is to ensure that sensitive data are not disclosed in an unauthorized manner while being transmitted between trusted hosts via trusted communication channels. Whenever the communication media is unprotected, encryption may be required for data confidentiality. The use of encryption is governed by the requirements of the data confidentiality policy, its supporting mechanisms, and the strength of the confidentiality protection

deemed necessary.

The requirements for data confidentiality are grouped into three areas: policy definition and enforcement, channel separation and protection, and cryptographic protection. The policy definition and enforcement requirements state that an organization's data confidentiality policy for data being transmitted across a communications channel should be defined and enforced. When physical and administrative means provide insufficient channel protection, all sensitive data items can be encrypted before transmission to provide the necessary protection. If data protection is based on encryption, the data confidentiality policy defines the mode of encryption and the encryption algorithm to be used. For a greater degree of protection, any data item, structure, or protocol control information that is exempt from this data confidentiality policy can be separated through the use of system privileges. Also, for some organizations, the establishment of thresholds for data confidentiality is important. That is, the leakage of sensitive data via channel bypass data (e.g., protocol control information) should not exceed a policy-specified threshold (i.e., the allowable bypass rate), and this threshold can be specified and enforced for all the communication protocols and channels supported by the distributed system.

Channel separation and protection requirements state that the distributed system must have the capability to protect the confidentiality of individual messages (e.g., requests, replies, commands) and selected control fields (e.g., sender and receiver identities, timestamps). Also, the degree to which channels are protected by physical and administrative means as opposed to how they are protected via encryption should be stated. Physical protection has to ensure that the compromise of data confidentiality is not feasible as a consequence of tampering with, or damage to, communication processors and media. Protecting the confidentiality of individual messages also involves how these messages are routed; It may be desirable to restrict the routing (the network links) of the channel data to secure communication media (e.g., network links that are physically protected). Another degree of protection can be provided by having separate communication channels for selected policy attributes.

Whenever physical and administrative means provide insufficient channel protection, cryptographic protection may be required. As stated, an organization chooses a cryptographic algorithm based upon national, international, and organizational standards. It may also be important for an organization to utilize different cryptographic algorithms for different protocols (e.g., mail or interprocess communication data), and for different policy attributes. Data confidentiality can also be enforced on specific types of communication, such as sensitive but unclassified data.

3.2 Data Integrity

The goal of data integrity is to ensure that message data are not modified in an undetectable manner while being transmitted between the trusted hosts of a distributed system via communication channels. Satisfying this goal also ensures that the source that created the message originally is unmodified and, therefore, becomes known to the message recipient.

Whenever the communication media is unprotected, cryptography (e.g., checksums and/or digital signatures) can provide for data integrity. The use of encryption is governed by the requirements of the data integrity policy, its supporting mechanisms, and the strength of the integrity protection deemed necessary.

The requirements for data integrity are grouped into three areas: policy definition and enforcement, channel separation and protection, and cryptographic protection. The policy definition and enforcement requirements state that an integrity policy for data being transmitted across a communications channel must be defined and enforced. This policy defines the scope of integrity protection, selects the integrity check functions to be used, and identifies any replay detection functions (e.g., functions based on sliding time windows and replay buffers, sequence numbers, random numbers, or combinations thereof) for messages or message streams. Cryptographic checksums and digital signatures can be used to ensure that the integrity policy is preserved over the lifetime of the secret key. In particular, without the knowledge of secret or private keys, it must be computationally infeasible to derive a signature or checksum for a plaintext message, and to derive a plaintext message from a signature or checksum. Also, for some organizations, the establishment of thresholds for data integrity is important. That is, for each protected channel and protocol, the risk that any illegitimate (e.g, modified or replayed) message or message stream is accepted as legitimate by a recipient after the integrity check functions and replay detections functions have been employed should be less than a specified threshold. Integrity check functions allow for the detection of any:

- modification and/or substitution of a message data item or of a stream,
- change in the order of a message data item or stream, and
- change in the number of message data items or streams.

Replay detection functions allow for the detection of the replays of old messages, message streams, or any parts thereof.

In the area of channel separation and protection, the distributed system must have the capability to protect the integrity of individual messages (e.g, requests, replies, commands) and selected control fields (e.g., sender and receiver identities, timestamps). As with data confidentiality, the degree to which channels are protected by physical and/or administrative means as opposed to how they are protected via encryption should be described. Physical protection has to ensure that the compromise of data integrity is not feasible as a consequence of tampering with, or damage to, communication processors and media.

As stated, cryptographic protection (e.g., checksums and/or digital signatures) may be utilized whenever physical and administrative means provide insufficient channel protection. Also, organizations may wish to configure their systems so that the data integrity functions use different checksums or signatures for different protocols (e.g., mail or interprocess communication data) or for specific types of data (e.g., sensitive but unclassified data). For a greater level of protection, organizations may also selectively allow for or mandate the use of encryption by, for example, assigning system privileges to different subject policy attributes.

3.3 Cryptographic Functions

As with the requirements for data confidentiality and integrity above, whenever the physical protection of the communication channels in a distributed system is inadequate or the communication channel cannot be protected by administrative means, cryptography is one way of providing the necessary means to implement channel separation and data protection. The requirements for cryptographic protection specify both policy and mechanism. Cryptographic policies are generally more extensive than access control policies. This is the case because cryptographic policies refer both to the cryptographic policies designed into the system (e.g., key management policies including those of key generation, installation, distribution, import/export,

activation, maintenance, destruction, escrow, and use), and to the cryptographic function configuration (e.g., 40 bit DES keys instead of 56 bit keys for some applications or products) and selection (e.g., cryptographic algorithm "A" must be used instead of algorithm "B" for top secret data). Cryptographic policies must also be defined relative to the data confidentiality and integrity policies stated above. For example, if the data confidentiality and integrity policies establish a threshold "T" for breaches of data confidentiality and integrity, the threshold established by the cryptographic function for key secrecy cannot be lower than "T." That is, there are policy dependencies between the cryptographic policies and the policies for data confidentiality and/or data integrity. The goals of the cryptographic functional requirements are:

- the specification of a cryptographic function of appropriate strength and of algorithms to support it,
- the protection of the cryptographic domain, and
- the secure management of keys within a product.

The first goal is important because crypto-analytic attacks that attempt to discover secret keys used by these functions can be mounted against most functions of a product both by external intruders and by legitimate users. The cryptographic requirements are also important because, as shown in the previous sections, they can be relied upon by several other components, such as data confidentiality, data integrity, and also by distributed identification and authentication, which are all the basis for trusted channel support in both centralized and distributed system products.

The second goal is important because the security of the cryptographic functions can only be provided if the cryptographic domain is resistant to external interference and tampering. The cryptographic domain executes the cryptographic algorithms in hardware, microcode, and/or software. It uses the secret key in plaintext form and maintains the key configuration options, initialization data, and key storage. Breaches in the cryptographic domain would be particularly dangerous since they can potentially affect the security of all system users and trusted hosts beyond the boundaries of a single product.

The third goal is important because the management of secure keys often provides the weakest link in the chain of cryptographic function mechanisms and use. The generation of poor-quality keys, inadequate key distribution, ineffective administrative procedures for key installation, weak key protection in storage, lack of limited key lifetime enforcement, and incorrect separation of keys, can lead to real security breaches.

The requirements for secure cryptographic functions are grouped into three areas: secure cryptographic function definition, cryptographic domain protection, and secure key management.² The requirements for secure cryptographic functions state that the cryptographic function, and the secret or private key space and lifetime, should be chosen so that the risk of unauthorized key discovery is within the limit determined by the system security policy. During the lifetime of the cryptographically protected data, an exhaustive search that discovers the secret or private key that was used should be computationally infeasible. The cryptographic function also ensures that

² The requirements for the physical security of the cryptographic domain and for the operational assurance for key installation are compatible with the Federal Information Processing Standard (FIPS) for Security Requirements for Cryptographic Modules [FIPS140-1], and the reader should refer to this FIPS for requirements in these areas.

the mapping from ciphertext to plaintext is such that, given an element of ciphertext, the computation of the corresponding element of plaintext, and vice versa, is infeasible. As stated in the data confidentiality and integrity components, the cryptographic algorithms that are selected should be in accordance with international, national, and industry standards.

Minimally, the cryptographic domain of each host must be protected by the trusted computing base (TCB) of that host. By definition, a TCB is isolated and non-circumventable (tamperproof). For a greater degree of protection, the cryptographic domain of each host can be a logically separate and distinct subset of the TCB domain of that host. An organization can also choose a more stringent physical requirement for a separate cryptographic domain to ensure that a compromise of the secret or private keys is not possible as a consequence of physical tampering with, or damage to, the host and/or cryptographic domain.

Keys must also be managed securely. Therefore, organizations can select among the requirements for key generation, installation, and distribution. The key generation process ensures that the secret key being generated is unpredictable. Key installation should be performed using a protected function (e.g., a trusted path such as a smartcard-based trusted path). When a key is distributed, the key should only be distributed to authenticated subjects. The key distribution process maintains the key's protection, establishes that the key is not an unauthorized replay, and establishes the set of subjects that is able to use the key. Also, key management controls the appearance of plaintext key values and the archiving of keys. Plaintext key values must only appear within the cryptographic domain, and must never be accessible outside of this domain. When a key is not in use, the key should be archived in encrypted form and stored in a physical location where it is protected from unauthorized disclosure, modification, substitution, or use. For a greater level of protection, separate, independent keys can be defined for each type of cryptographic function. For example, an organization may require one key for the encryption and decryption of data and a separate key for authentication.

For an organization with a key escrow policy, this policy must be defined and enforced. A key escrow policy must state:

- the type of keys to be escrowed,
- the global identifiers used for key identification,
- the binding of the key to the subjects using that key,
- the escrow period,
- the escrow authority, and
- the procedures for accessing the encrypted key within the escrow facility.

3.4 Distributed Identification and Authentication

The role of the identification function in a secure distributed system is to assign a unique, unambiguous name or identifier to all subjects (e.g., users, realms, communication channels) that perform any action that should be mediated within the system. For example, there should be a capability of associating unique user identifiers with all auditable actions taken by an individual. The role of the authentication function is to attribute responsibility for an action to an identified subject. All users are required to identify and authenticate themselves before beginning any actions that must be mediated. These actions are typically invoked by requests on a channel. For example, user authentication involves the verification of the user identity claimed during a login request on a login channel. For other types of channels, the authentication function simply

determines the identity of the subjects at one or both ends of a channel.

The requirements for distributed identification and authentication fall into five areas: channel identification, channel authentication, user authentication, cross-realm authentication trust, and channel authentication policy. The requirements for channel identification state that all types of simple subjects that must be authenticated on the channel should be identified. A *simple subject* can be a process, a group of users, a host, a communication channel, a realm, a service, or a program. These simple subjects are authenticated based on the distributed system's security policy. The identification of each of these subjects should be:

- complete: all users and subjects, including privileged subjects, must be identified, and,
- unambiguous: every user and every subject must have a different identity, and this identity cannot be reused.

A secure distributed system can also allow communication between compound subjects. A *compound subject* is a concatenation of one or more subjects. These can include delegation chains, restricted delegation chains, and conjunctions of subjects (AND-chained identities), as required by the system security policy. A *delegation chain* is a list of subjects that are acting on behalf of one particular subject. When a subject becomes part of a delegation chain, the subject decides whether it should enable or disable the delegation of its identity. A *restricted delegation chain* gives each subject the capability of restricting the delegation of its authority to another subject by restricting its delegation to only a subset of its policy attributes. With any type of delegation chain, the distributed system must be able to preserve the distinction between the identity of the original subject and that of its delegates. The distributed system must be able to confirm that each subject accepted this delegation and the delegation of some or all of its policy attributes. An *AND-chained identity* can be used to identify subjects as they pass from one realm to another. Each subject in either delegation chains or AND-chained identities are required to be authenticated individually.

For channel authentication, when a subject receives a message it should be able to ascertain the channel on which the messages arrives, and the direction of the message on the channel. The distributed system can also perform *message-origin authentication* on the channel, that is, whenever a subject receives a message on a channel, it can ascertain which subject originally created that message. The distributed system can also have the capability to perform *mutual authentication* on a channel, that is, whenever two subjects exchange messages with each other, each recipient can know that the received message was originally created by the other subject as part of that message exchange. Both message origin and mutual authentication establish that these channel messages are not replays of messages that had originated earlier. The validity of channel authentication can be limited to the duration of the channel key lifetime or to specific intervals of validity.

User authentication functions protect the authentication data that verifies the identity of individual users (for example, passwords, seeds, and secret keys) to prevent one user from masquerading as another. The exposure of this data should be minimized to decrease the possibility of unauthorized disclosure, modification, deletion, substitution, or use. If sharing of authentication data among trusted hosts is required, then, as shown in Figure 1, this sharing can be minimized by the utilization of authentication authorities that act as trusted third parties so that data can be shared on a pair-wise private basis. Through the use of authentication authorities, user authentication functions can support single user login regardless of the number of realms or per

realm hosts in the distributed system.

If a distributed system is partitioned into one or more separate administrative realms, authentication paths among the authorities of multiple realms should be defined and enforced in accordance with the defined policy for inter-realm authentication. In that case, subjects have to be able to discover the identity of the trusted authorities. In a multi-realm distributed system that has traveling users, a traveling user should be able to authenticate him/herself (that is, should be able to login) in a foreign realm in accordance with the inter-realm authentication policy.

In addition, all security policies supported by the authentication functions should be defined and enforced. Each policy should specify:

- the types of subject and types of authentication supported (for example, types of channel and user authentication),
- the time and duration of channel authentication (for example, login session, RPC bind, call, and packet authentication) and the revocation conditions, and
- the validity and renewability of authentication data.

4.0 Future Work

The total task force report detailing the security requirements for distributed systems that were developed is scheduled to be released as a NIST Internal Report (IR) in the fall of 1995. The Common Criteria, which is scheduled to be released in January of 1996, will address secure distributed system requirements for both the functional and assurance areas. Future plans call for work on the mutual recognition of evaluations for products/systems that are created using the Common Criteria.

References

- [COMCRIT] *Common Criteria for Information Technology Security Evaluations, Rationale, Parts 1, 2, and 3, Version 0.6, April 1994.*
- [CTCPEC] *Canadian Trusted Computer Product Evaluation Criteria, Communications Security Establishment, Version 3.0e, January 1993.*
- [DBI] *Trusted Data Base Management Systems Interpretation of the Trusted Computer Systems Evaluation Criteria, NCSC-TG-21, Version 1, April 1991.*
- [FEDCRIT] *Federal Criteria for Information Technology Security, Volumes I and II, NIST/NSA, Version 1, December 1992.*
- [FIPS140] *Security Requirements for Cryptographic Modules, FIPS PUB 140-1, Draft 1993 May 24, NIST.*
- [ITSEC] *Information Technology Security Evaluation Criteria, Commission of the European Community, Version 1.2, June 1991.*
- [TCSEC] Department of Defense, *Trusted Computer Security Evaluation Criteria, DOD 5200.28-STD, 1985.*
- [TNI] *Trusted Network Interpretation of the Trusted Computer Systems Evaluation Criteria, NCSC-TG-005, Version 1, July 1987.*

A PERSPECTIVE OF EVALUATION IN THE UK VERSUS THE US

Alan Borrett

**Member of UK ITSEC Scheme,
P.O. Box 152, Cheltenham, Glos.
01242/221491 Ext. 4557.**

This paper identifies differences between TPEP evaluations and those which are carried out under the auspices of the UK ITSEC Scheme, together with aspects that the author personally likes about each evaluation process and recommendations for how they may be more closely aligned. It is hoped that it will be of value to vendors entering evaluations in the UK and US, and to those involved in Common Methodology.

6 June, 1995.

Product evaluations in the United Kingdom (UK) and the United States (US) are the way they are because of a number of factors: criteria; evaluation process requirements, such as quality and consistency; evaluation resourcing. This paper is written from the perspective of a former UK Information Technology Security Evaluation and Certification (ITSEC) Scheme certifier who has also worked as a Trusted Product Evaluation Program (TPEP) evaluator for some time. It contrasts differences between TPEP and the UK ITSEC Scheme, identifies aspects of each which the other may benefit from, and presents proposals for bringing the two evaluation processes closer together. It is hoped that this paper will be of value to vendors who will be supporting evaluations in both countries and as such will need to be party to the full implications, and for those involved in the production of Common Methodology.

Historical Perspective

UK evaluation criteria has evolved from criteria which was developed for system evaluations. UK system criteria was subsumed by Information Technology Security Evaluation Criteria [ITSEC], which is equally applicable to systems and products. ITSEC is supported by the Information Technology Security Evaluation Manual [ITSEM]. A series of UK computer security advice and ITSEC and ITSEM interpretation documentation exists in the form of UK security publications [UKSP01 to UKSP05 and UKSP07]. In advance of the UK system criteria and contrary to the UK system approach, the US developed evaluation criteria for products (Trusted Computer System

Evaluation Criteria [TCSEC]). The TCSEC was subsequently interpreted for specific technology and environments in the form of the accompanying "Rainbow" series of documents.

Criteria

Evaluation criteria has largely been responsible for divergence between the US and UK approaches to evaluation. Furthermore, criteria combined with process has resulted in evaluators performing security analysis and testing in the US, while in UK ITSEC Scheme evaluations, the onus for security analysis and testing is placed on the developer, and UK evaluators independently review and audit the work of the developer and report their findings according to the requirements of criteria and methodology. Effectiveness is another area where the US and UK evaluation processes differ. Effectiveness analysis is implied in TCSEC, was carried out when the TCSEC was established, and is performed by evaluators during pre-evaluation and to some extent during the initial stages of design analysis. Progression to evaluation only occurs when the evaluators are content that the product appears to be effective. In contrast, ITSEC effectiveness is performed during the evaluation stage. In UK evaluations, effectiveness is covered under Evaluation (Phase 2). There is probably little scope for ITSEC and TCSEC criteria differences to be reconciled prior to Common Criteria.

Criteria - Testing Assurance

In the UK, the onus to carry out testing (with the exception of penetration testing) to a suitable level rests with the vendor or developer. Evaluators are required to witness the developer's testing, repeat some tests for themselves, and perform penetration tests and tests which search for errors. After evaluation of the developer's test plan, UK evaluators could recommend that the developer perform additional tests.

In the UK, the claims made in the Security Target become the target of ITSEC penetration testing strategy. For E4 to E6, the vendor has to supply a justification of sufficient test coverage. Penetration testing is applied between E1 and E6 inclusive. The degree of testing performed in UK evaluations is related to the assurance level: testing against the Security Target (for E1 to E6), the detailed design (for E3 to E6) and the source code (for E3 to E6). This amounts to module, integration and system testing being required at E3 and above. UK evaluators repeat some of the vendor's functional tests. Product testing is usually carried out at the Commercial Licensed Evaluation Facility (CLEF) site and system testing at the development or operational site. CLEF quality procedures comply with the National Accreditation Measurement Assessment Services [NAM10 and NAM11] requirements for test laboratories. The Certification Body provides input to the tests to be performed by the evaluators.

The US defines functional testing as aiming to show that the implementation meets the specification and penetration testing as an unconditional search for errors. The onus is on the vendor to produce a functional test suite. At B1 and below only functional testing is carried out; the US does not deliberately search for side effects but if any are found (with the exception of covert channels) they must be removed; the theory being that penetration testing flaws would be too prevalent at B1 and below, owing to minimal architectural assurance being available. All TCB external interfaces are

tested (both program and otherwise) together with procedures to bring the system into and maintain a trusted state. Internal TCB interfaces and the trusted subject interfaces are not tested but the US is moving towards testing them because it eases the task of subsequent application evaluation. Both positive and negative testing is carried out and common trouble areas exploited; negative testing is subjective. In order to get the most from testing assurance, the entire set of test suites is expected to have been successfully run by the vendor. In addition to carrying out new tests, the evaluators repeat the vendor's entire set of test suites. Testing is performed on a configuration which is representative of the product which has undergone security analysis as configured by the evaluation team. This usually takes place at the vendor's site on the vendor's equipment.

The author suggests that there is room for meeting half-way on testing: North American evaluators should increase their test coverage and Europeans should reexamine test requirements, possibly not introducing penetration testing until E2, with module and system level testing possibly sufficing. Common Methodology should define the testing responsibility of developers and evaluators.

Criteria - Covert Channel Analysis

The UK consider covert channels to be a minor threat in practice owing to the difficulty of introducing malicious code into systems. The UK believes that two types exist: those which require malicious code and or co-operation with the sender and those which do not require co-operation. The type of covert channel of concern should be threat-related. ITSEC evaluations always address covert channels under effectiveness analysis and a determination is made based on the grounds claimed in the Security Target. This translates into covert channel issues being considered from E1 to E6 and for covert channel analysis to be covered more thoroughly at E3 and above.

The TCSEC definition of a covert channel only applies when there is a MAC policy. Covert channels are flaws which result from the sharing of resources and can never be completely avoided. TCSEC defines two types of covert channels: storage and timing channels. Arbitrary bandwidth limitation requirements are followed. The requirements for storage channels are more stringent than those for timing channels. This also leads to vendors attempting to classify covert channels as timing channels rather than storage channels. Trusted Product Evaluation Program (TPEP) can tolerate some covert channels but flaws have to be fixed. This invariably leads to vendors attempting to categorise flaws as covert channels. TPEP determines the acceptability of a covert channel based on its bandwidth but due regard is given to the type of information that is transmitted in non-TPEP evaluations; for example, in an Information Security (INFOSEC) system evaluation, a cryptovvariable could be transmitted over a channel of low bandwidth. The developer is required to calculate the theoretical maximum bandwidth of all exploitable covert channels. The calculation of bandwidth is an estimate. The following factors are taken into account: sample size, modulation rate, performance measurements, input/output speed, etc. US product evaluations at B2 and beyond, take into account multi-instantiations of covert channels.

Criteria - Architecture

In UK evaluations, evaluation is carried out against the Security Target. Should the Security Target include the underlying operating system and hardware, it is included within the scope of the evaluation, otherwise assertions are documented. The UK normally only look at hardware in the event that Security Enforcing Functions are specifically implemented in it, or by virtue of it being special purpose (not commercial off-the-shelf). Developers may include statements about platform independence in their Security Target; this will normally lead to caveats in the Certification Report. In other words, ITSEC excludes some architectural differences. Source code is a required ITSEC deliverable for E3 and above evaluations: beginning at E3, source code is used in correctness analysis when evaluators are required to sample code for traceability to design and test coverage analysis, and from a quality perspective, for adherence to vendor coding standards; source code is used in effectiveness analysis at E4 and above. TCSEC requires architectural evaluation down to the hardware level. By way of example, an application evaluation demands that the underlying software be included in the evaluation. US evaluators sample undefined opcodes and search for obvious flaws. For B1 and below evaluations, specifications are required for the hardware components and how the hardware components are bonded together. Evaluators assume that the specifications are accurate. The operating system together with the hardware are evaluated in combination. For higher-level TCSEC evaluations, architectural conformance standards are required. More emphasis is placed on undefined opcodes, trusted distribution and conformance testing. Code correspondence to design, adherence to coding standards and in-line commenting is less important than modularity, complexity, coupling, cohesion and redundancy at B2.

The bottom line is that the US appears to do much more hardware analysis and hardware testing than the UK. The UK appears to do more in the way of source code analysis for mid-range assurance levels.

Criteria - Developer Environment Assurance

The UK places importance on the developmental assurance aspects of products by third party evaluation. This covers physical security, clearances of developmental staff, integrity of developmental computers and quality and developmental procedures, to name but a few. ITSEC demands that Configuration Control and Trusted Distribution be in place for E1 to E6 inclusive, with the rigour increasing with assurance level. Developmental assurance aspects are not specifically considered by the US during evaluation. The onus is placed on evaluator verification of the product by design analysis and testing and the integrity of the developmental environment is not a consideration. Aspects of UK development environment assurance requirements can become visible during RAMP. TCSEC does not require Configuration Control and Trusted Distribution until B2 and A1 respectively.

It would be reasonably inexpensive for US vendors to adhere to developmental assurance requirements. In most cases, the existing procedures which vendors have in place to protect their own proprietary interests are sufficient to meet UK developmental assurance requirements.

Criteria - Developer Documentation Requirements

In ITSEC evaluations there is a relationship between the level of detail required of design documentation and the assurance level. This ranges from "starting to describe" to "explaining". UK evaluators ensure that traceability exists through the various levels of Target of Evaluation (TOE) representation. The onus is on the developer to provide traceability evidence. UK ITSEC documentation for trusted products comprises: security target; security policy model; effectiveness documentation, such as suitability analysis, binding analysis, strength of mechanism analysis, vulnerability analysis (both constructional and operational) and ease of use analysis; architectural detail; detailed design; testing; configuration control. TPEP evaluation teams study detailed architectural design documentation and establish traceability between design and functional tests. Vendors provide traceability between high-level design and test matrix documentation and the respective detailed design and test documentation. US trusted product documentation comprises: security policy; security policy model; philosophy of protection; TCB interfaces; architectural design; detailed design; test scenarios and evidence; details of configuration management; proposals for RAMP.

Visualising a correspondence between those documents required by ITSEC and those required by TCSEC is not straightforward at the more abstract levels of documentation. It is thought that the terms of reference of the US Philosophy of Protection could be extended to include effectiveness aspects. These aspects should be resolved within Common Criteria.

Evaluation Process

The UK Evaluation Methodology defines the work of evaluator, certifier and developer. The evaluation process comprises four phases: 1) Phase 0 - Pre-Evaluation Consultancy (e.g. deliverables list and advice); 2) Phase 1 - Preparation (e.g. evaluation work programme and detailed deliverables list); 3) Phase 2 - Evaluation (e.g. analysis of design and testing); 4) Phase 3 - Certification (including the final Evaluation Technical Report and Certification Report). The UK does not differentiate between the way in which system and product evaluations are carried out: both are performed against the same criteria and to the same methodology. The UK ITSEC Scheme allows for a rich mix of products to be evaluated: aside from traditional products such as operating systems, database products and personal computer security products, the UK has also evaluated commercial communications devices (although their sale to the private sector is prevented if the encryption element employs a government algorithm). The UK work to a documentation set which defines procedures for the Scheme. In addition, it will shortly adhere to a quality manual. Consistency between UK ITSEC Scheme evaluations is primarily the responsibility of the Certifier; although the UK's National Physics Laboratory also plays a part in accrediting CLEFs to perform evaluations and subsequently carrying out periodic and random audits of their work. Also, CLEFs work to a quality system which is in keeping with that of their parent company. Two certifiers and a member of the Certification Body responsible for evaluation methodology, review critical documents such as evaluation work programmes, evaluation technical reports and certification reports.

The new TPEP process [TPEPRO] comprises two stages: Advice and Evaluation. Advice is the term given to what was formally swept up by the Vendor Assistance Phase; Advice is normally complete by the time that the Initial Product Technical Report is produced. In the spirit that evaluators provide independent assurance that a product achieves its security features and assurance attributes, it would probably not be too difficult to produce an amalgamated evaluation process: Phase 0 and 1 of the UK approach would approximately equate to Advice under TPEP and all for intent and purpose, Phase 2 and Phase 3 would equate to TPEP's Evaluation. However, as the underlying evaluation philosophy is different, it is not easy to see how this aspect can be easily reconciled.

If confusion is to be avoided, it will be necessary for the US and UK to use the same evaluation terminology. Common Evaluation Methodology should address evaluation process alignment. European evaluators should be obliged to express and defend their understanding of how security is enforced within products; North American evaluators will have to work to a documented evaluation methodology and quality system.

Evaluation Process: IPTR

The concept of an Intensive Preliminary Technical Review (IPTR) does not exist in the UK. UK evaluators review a vendor's ITSEC deliverables for ITSEC compliance and raise problem reports on any shortcomings. An IPTR is conducted by the US to determine a vendor's readiness for evaluation. A move into Evaluation is commenced should the IPTR prove to be satisfactory. The IPTR precedes design analysis and testing.

In a commercial evaluation environment such as that in place in the UK, it is not deemed necessary for evaluators to produce documentation which justifies a vendor's readiness for evaluation. It is not apparent that the cost of UK IPTRs, which in turn would be passed to evaluation sponsors, would result in much benefit to the UK. Additionally, an IPTR in a commercial scheme could give rise to embarrassment for the government body responsible for approving it, should it subsequently be proven that the developer was not ready for evaluation. The author believes this to be one area where a divergence in evaluation process is not detrimental to reciprocity.

Evaluation Process: IPAR

Upon completion of design analysis, US evaluators describe their understanding of the security design of a product in a document entitled an "Initial Product Assessment Report" (IPAR). The IPAR is updated at the end of the evaluation and becomes the Final Evaluation Report. The IPAR has traditionally been written by evaluators; however, vendors can be invited to supply specialised sections and the evaluation team is required to defend it. It is on the basis of a successful IPAR and IPAR Trusted Review Board (TRB), that US evaluators are permitted to pursue testing.

The concept of evaluators having to convince a TRB that they understand how security works in a product does not exist in the UK. The introduction of IPAR production in the UK would introduce an additional overhead for evaluators and certifiers: evaluators and senior evaluators respectively, would

be responsible for writing and reviewing it; certifiers would also find themselves with an additional document for review.

The author is very much in favour of an IPAR and at the risk of placing an extra burden on UK evaluators and certifiers, suggest that it be adopted in the UK. It may be possible for CLEFs' Evaluation Work Programmes to be extended to include the content of a US IPAR. The IPAR should then be reviewed by a UK TRB.

Evaluation Process: TRBs

Government oversight of evaluations differs between the UK and US. Evaluation oversight in the UK is essentially a peer review process: CLEF evaluators work to quality schemes imposed by their parent company and the Certification Body; the work of evaluators is reviewed by other evaluators within the same CLEF and the Certification Body. The concept of a Technical Review Board (TRB) does not exist in the UK. TPEP consistency is maintained by the Technical Review Board. The TRB accepts presentations by the evaluators of their IPAR (which is intended to convey evaluators' understanding of how security is enforced by the product), the evaluation team's test proposals and the Final Evaluation Report. Satisfactory TRBs are required before the evaluation can proceed to the next stage. In addition to ascertaining the knowledge that evaluators have of the product which they are evaluating, the TRB also serves as a training forum for new evaluators. Assuming that European TRBs will be necessary, their composition should comprise senior evaluators and the certifier. In order that senior evaluators provide the best added value to a TRB, they would have to retain a working evaluation presence. Conflicts between evaluation confidentiality and TRB work would have to be carefully thought through. At the risk of placing an undue overhead on North American TRBs, an alternative to a European TRB would be for European products to be presented at North American TRBs. This may be a more cost effective option in the short term, however, timing of TRBs could be critical, an undue overhead could be placed on the extant TRB process and travelling costs could mean that it would only be marginally cheaper.

If North American confidence is to be had in European evaluation processes and vice versa, the author believes that TRBs are likely to be the linch-pin of any reciprocity agreement. In any event, there would be some mileage to be had by US TRB members serving on selected UK TRBs and vice versa.

Evaluation Process: Evaluator, Certifier and Vendor Consistency

Joint Technical Reviews are held for evaluators in the UK; they are of a day's duration and it is optional whether evaluators attend. UK evaluators are obliged to attend evaluator training modules. These address how evaluators should work and how the criteria and methodology should be applied to evaluations; lessons learnt from past evaluations are also presented. UK evaluators are graded according to the courses they have attended and the evaluation experience they have attained. A common electronic information processing and storage system does not exist between CLEFs, the Certification Body and Scheme vendors. Consequently, information exchange for the better part takes

place in hard-copy format. In the UK, a combination of ITSEC, ITSEM and the Scheme has meant that government oversight has taken on a quality perspective, with delegation for understanding security principles and concepts to the CLEFs.

The US places importance on technical computer security within the US evaluation community. It has a thorough INFOSEC training programme. The majority of courses have three components to them: lectured material supplemented by detailed handouts; class exercises; formal examinations. Class exercises really get the message across and the examination is intended to assess how much material students have absorbed. Annual evaluation and twice-yearly vendor workshops assist in maintaining evaluation and developer consistency in the US. Each workshop is of several days duration. Evaluator attendance is mandatory at evaluation workshops. The National Security Agency (NSA) runs in-house evaluator training which lasts an hour a week; it is left up to evaluators whether they attend. Electronic information exchange for all TPEP participants exists in the form of Dockmaster. The Dockmaster Interpretations forum provides a mechanism for evaluation teams to discuss and obtain comments from other evaluators on the interpretation of TCSEC, while still observing proprietary protocol. Dockmaster also contains product evaluation history information.

It is felt that European evaluators and certifiers will have to convince the US that they are technically security aware as well as quality aware. This could be best achieved by certifiers working as evaluators for a time and a formal education system being in place for both evaluators and certifiers. Europe should develop evaluator and vendor workshops along the lines of the US; it should be mandatory that all evaluators attend. Training modules should cover the likes of formal methods, networking security, database security, operating system security, communications protocols, UNIX™ security, security architectures and compartmented mode workstation security. European certifiers should receive training on the products that they are responsible for certifying as a matter of course.

Evaluation Process: Evaluation Tools

UK ITSEC evaluations do not demand that UK supported tools be used. A suitable tool, fit for purpose, is required. For high assurance products the US maintains endorsed tools: Gypsy Verification Environment; Formal Development Methodology; Hierarchical Development Methodology. These are contained on the Endorsed Tools List (ETL). The UK has encouraged Z and CSP for formal specifications and MALPAS and SPADE as source code verification tools.

Reciprocity will probably dictate that the US ETL include appropriate European tools. North America and Europe should seek to develop a harmonised approach to the grading or rating of evaluation tools. North America and Europe should seek to jointly operate and maintain a list of evaluation tools.

Evaluation Process: Sources of Vulnerability Information

The UK maintains a database of both public-domain vulnerabilities and those which have arisen from CLEF evaluations. It is the responsibility of the Certification Body to ensure that CLEFs have pertinent vulnerability information pertaining to products under evaluation. This combined with the vendor's vulnerability analysis helps to ensure credibility of the commercial Scheme. The US derives product vulnerability information from the following sources: previous evaluations; public vulnerabilities; design analysis. However, an organised collection of vulnerability sources does not exist for TPEP.

Interchange of vulnerability information between Europe and North America should be envisaged. This may have a bearing on the type of database which is used. The sharing of both product and system vulnerability information should be pursued; however, the sensitivity of system vulnerability information may impose some constraints.

Evaluation Process: Maintenance of Evaluated Status

During UK evaluations, the design and implementation are labelled according to their effect upon the security policy of the product or security target, according to the following: security-enforcing, security-relevant, non-security relevant. The TCB would comprise security-enforcing and security-relevant; it may be possible to make security-relevant changes without calling into question the certified status. In the case of non-security-relevant changes, the change to the target of evaluation could be effected without affecting the status of the certificate.

The US has Rating Maintenance Phase (RAMP) in which changes are categorised on a scale of A to H. A complete re-evaluation is required for H whilst for A the vendor merely has to mail the analysis to the evaluators. US developers combine products produced by different vendors. Combinations of products are not currently evaluated by TPEP, however, specific combinations are analysed. When it comes to re-evaluation, NSA reuses evaluation results by using as many of the previous evaluation team as possible to carry out the re-evaluation.

The UK approach to classifying the composition of a product's internals according to their effect on its security policy, should be adopted in the US because it assists in re-evaluations. The UK should allow some security responsibility to be given back to the vendor by instigating a UK RAMP scheme; it would seem sensible to adopt the US scale of A to H for changes. [Note: the UK are currently investigating a UK ratings maintenance system.]

Liability

In addition to technical differences which have arisen from criteria, process and oversight, there are legal liability issues to be resolved. For instance, German law demands that someone be liable for failures in certified products. The US makes specific explicit disclaimers to the effect that it assumes no responsibility and therefore the customer is at risk. The imminent mutual recognition of

certification results agreement between the UK and Germany has ramifications for any future mutual recognition agreement between the US and UK; it could be inferred that the UK would be held accountable for any deficiencies which are subsequently found in US evaluated products which appear in the UK Certified Product List [UKSP06]; alternatively, the liability issue may pass to the US, which at this point in time, is protected from litigation by virtue of being a US government body.

The political implications of legal liability for Europe and North America merits further investigation. In the interim, it may suffice to place an appropriate caveat alongside any US evaluated products which appear in UK Certified Product List publications.

Quality of Evaluations

Even though the UK require that all techniques and lessons learnt from evaluations be documented at the end of an evaluation and made available to the UK evaluation community, it is felt that CLEFs prepare this information from a position of non-disclosure of information which is of proprietary interest to them. There is concern in the US that UK evaluations, by virtue of their commercial nature, do not encourage the sharing of evaluation techniques amongst the evaluation community. This arises from it not being in the best interest of specific CLEFs to share information with their competitors. This issue does not feature in the US because all evaluators are funded from federal resources. There is no doubt that US evaluations have become more efficient over time owing to an unimpeded interchange of evaluation information. It is difficult to see how a similar free-flow of benefits to the UK evaluation community can be achieved. The effect of this matter on reciprocity requires further investigation.

Common Methodology

The European Community has invested heavily in evaluation methodology. The intent has been to achieve repeatability and reproducibility of evaluation results, and since all European evaluations are funded by a sponsor, to ensure that the quality of evaluation is not degraded by commercial pressures. Evaluation methodology has created a quality platform and basis for consistency, and defined the role of evaluator and developer, together with that of those responsible for overseeing evaluations. This in turn has meant that there is now no scope for evaluators to compensate for the work of developers, for example, by carrying out source code analysis to make up for deficiencies in design documentation. UK evaluators would be obliged to report the problem and suspend that aspect of the evaluation affected by the problem, until the developer carried out remedial action. It is felt that documented Common Methodology will be required for reciprocity. An appropriate forum for criteria interpretations may be Common Methodology. Common Methodology should therefore be jointly maintained between Europe and North America. Some thought should be given to the role of the European Commission (EC) in methodology. As part of the wider context, North America will have to develop adherence to a formally documented evaluation methodology. North America and Europe should actively seek to harmonise their evaluation approaches to a common methodology. North America and Europe should seek to jointly operate and maintain such an evaluation methodology.

	US	UK
Evaluation Consistency	Technical Review Boards Dockmaster Thorough Compusec Training Evaluation Workshops Initial Product Assessment Report	Adherence to Methodology Methodology Training
Maintenance of Evaluation Status	RAMP	Re-evaluation
Funding	Government Resourced	Sponsor Resourced
Training	Comprehensive Compusec Courses Weekly in-house Evaluator and Vendor Workshops (All evaluators are expected to attend Evaluation Workshops)	A Couple of Weeks of Methodology Joint Technical Reviews (Representatives of the Evaluation and Certification Communities attend)
Criteria	TCSEC CMWREQ and CMWEC	ITSEC
Evaluation Readiness	Intensive Preliminary Technical Review	Contractual (Suspend evaluation if found not to be ready)
Evaluator Responsibility	Perform Security Analysis	Audit and Review the Security Analysis of the Developer
Evaluation Tools	Dockmaster Gypsy HDM FDM	Z (BALZAC) MALPAS SPADE

Summary of Process Differences

Glossary

Assurance: the confidence that may be held in the security provided by a Target of Evaluation.

Correctness: a property of a representation of a Target of Evaluation such that it accurately reflects the stated security target for that system or product.

Covert Channel: the use of a mechanism not intended for communication to transfer information in a way which violates security.

Developer: the person or organisation that manufactures a Target of Evaluation.

Effectiveness: a property of a Target of Evaluation representing how well it provides security in the context of its actual or proposed operational use. [For information concerning the work that evaluators and developers are expected to perform in respect of effectiveness analysis, the reader is referred to Annex B of: Manual of Computer Security Evaluation, Part II, Standard Evaluation Work Programmes, UKSP05, Issue 1.0, dated 14 December 1994.]

Evaluation: the assessment of an IT system or product against defined evaluation criteria.

Evaluator: the independent person or organisation that performs an evaluation.

Integrity: the prevention of the unauthorised modification of information.

Opcode: the instruction level equivalent in assembly language.

Penetration Testing: tests performed by an evaluator on the Target of Evaluation in order to confirm whether or not known vulnerabilities are actually exploitable in practice.

Product: a package of IT software and/or hardware, providing functionality designed for use or incorporation within a multiplicity of systems.

Security Enforcing: that which directly contributes to satisfying the security objectives of the Target of Evaluation.

Security Mechanism: the logic or algorithm that implements a particular security enforcing or security relevant function in hardware and software.

Security Policy: the rules and regulations governing the handling of information.

Security Relevant: that which is not security enforcing, but must function correctly for the Target of Evaluation to enforce security.

Security Target: a specification of the security required of a Target of Evaluation, used as a baseline for evaluation. The security target will specify the security enforcing functions of the Target of Evaluation. It will also specify the security objectives, the threats to those objectives, and any specific security mechanisms that will be employed.

System: a specific IT installation, with a particular purpose and operational environment.

Sponsor: the person or organisation that requests an evaluation.

Target of Evaluation: an IT system or product which is subjected to security evaluation.

Vulnerability: a security weakness in a Target of Evaluation (for example, due to failures in analysis, design, implementation or operation).

References

- ITSEC Information Technology Security Evaluation Criteria (ITSEC), Version 1.2, dated June 1991.
- ITSEM Information Technology Security Evaluation Manual (ITSEM), Version 1.0, dated 10 September 1993.
- UKSP01 Description of the Scheme, Issue 2.0, dated April 1994.
- UKSP02 The Licensing of Commercial Licensed Evaluation Facilities, Issue 2.0, dated 1 May 1995.
- UKSP05 Manual of Computer Security Evaluations, UKSP 05 Part I, Procedures to be followed by a CLEF in conducting evaluations under the Scheme, Issue 3.0, dated 1 October 1994.
- UKSP05 Manual of Computer Security Evaluations, UKSP 05 Part II, Standard Evaluation Work Programmes, Issue 1.0, dated 14 December 1994.
- UKSP05 Manual of Computer Security Evaluations, UKSP 05 Part III, Tools and Techniques to be used by a CLEF in Conducting Security Evaluations under the Scheme, Issue 1.0, dated 24 June 1994.
- UKSP05 Manual of Computer Security Evaluations, UKSP 05 Part IV, Lessons Learnt from Practical Evaluation Experience under ITSEC, Issue 1.0, dated 24 October 1994.
- UKSP07 Certifiers' Guide, dated September 1993.
- TCSEC DOD 5200.28-STD, Department of Defense Trusted Computer System Evaluation Criteria, dated December 1985.
- NAM10 M10 - NAMAS Accreditation Standard, General Criteria of Competence for Calibration and Testing Laboratories; Edition 1, dated March 1989.
- NAM11 M11 - NAMAS Regulations, Regulations to be met by Calibration and Testing Laboratories; Edition 1, dated April 1989.
- TPEPRO C71 Trusted Product Evaluation Program Process Document, dated 27 March 1995.
- UKSP06 Certified Product List, dated April 1995.

TM - all trademarks are acknowledged.

ECMA's Approach for IT Security Evaluations

Alexander Herrigel

r³ security engineering ag, Switzerland, Email: herrigel@r3.ch

Roger French

Digital Equipment Corporation, U.S., Email: French@ateis.enet.dec.com

Haruki Tabuchi

Fujitsu Ltd, Japan, Email: Tabuchi@Saint.NM.Fujitsu.co.jp

European Computer Manufactures Association (ECMA), TC36/TG1

Abstract

The increasing globalization of the commercial market forces multinational companies to purchase, install and run enterprise wide computer and telecom systems. Effective protection of these systems affects the IT security strategy of the enterprise. IT security evaluation criteria have to be investigated and selected for implementing an adequate protection. This paper presents ECMA's approach for IT security evaluations. In contrast to other standards, it is the objective of the Commercial Oriented Functionality Class (COFC) to specify only the minimum set of security functionalities for the commercial market to reduce technical complexity, and to allow the cost- and time effective application. In addition, the standard is based on today's commercial requirements and not on military and governmental requirements which are quite different. COFC is considered as a baseline standard commercial enterprises can measure against.

1. Introduction

The increasing globalization of the commercial market forces multinational companies to purchase, install and run enterprise wide computer and telecom systems. Since these systems constitute the basis for providing competitive services to the customers of the enterprise, different security aspects are of great significance for the enterprises business and its continuity. In addition, information management is critical to the competitive position of a company in a specific commercial market segment. The enterprise systems and the processed information are subject to a number of threats from different sources: Employees make mistakes, some have difficulties in system handling, a few commit fraud because of personal or commercial reasons. Outsiders or unauthorized persons may target an enterprise for fraud, vandalism or espionage. Effective protection of enterprise systems and information databases requires a structured management approach to security. This approach affects different domains of the enterprise such as people, computer or telecom equipment, communication networks, buildings, business and facility planing etc. . Typically, the Corporate Security Policy triggers a Risk Assessment which results in a Corporate IT Security Policy. Typically, a Corporate Security Policy requires a Risk Assessment covering at least the following aspects:

- What are the assets of the enterprise ?
- What is the value of these assets ?
- Who should have access to these assets ?
- Which threats have to be encountered for these assets ?
- What is the impact of failure ?
- How is disaster recovery and restart organized and managed ?
- What operational rules and which security enforcing procedures are needed to fulfill the Corporate Security Policy ?

The risk assessment and a following risk acceptance result in a Corporate IT Security Policy which has to be adopted by the enterprise management. Its realization affects the IT systems with respect to application security, computer or telecom systems security and network security. The installed systems must fulfill the imposed security requirements of the Corporate Security Policy. IT security evaluation criteria have to be selected for implementing an adequate protection. The specified security evaluation criteria form a decision base for purchasing a telecom or computer system from a specific manufacturer. The evaluated systems must fulfill the imposed requirements and constraints of the Corporate Security Policy. Typically, a specific operating system/hardware platform combination must support an adequate set of security enforcing functions with respect to the Corporate Security Policy.

2. Related Work

A number of standards or quasi standards are available or under development, which can be applied to evaluate a specific assurance level of security for a hardware / operating system combination. One is the somewhat dated and US government/DoD-centric TCSEC, the "Orange Book" [1]. Others [2 - 6] like the Information Technology Security Evaluation Criteria (ITSEC) [2], the Federal Criteria (FC) [5] or the Common Criteria (CC) [6] have or are being recently developed, including the 800 page Common Criteria which is a five government effort to provide a harmonized criteria for the U.S., Canada and the European Union. There is also the ISO 3 part criteria (ISO/IEC JTC/SC27/WG3) and several industry specific criteria. The CC and ISO standards are still under development. They need a complimentary evaluation scheme and mutual evaluation recognition to be of practical value. Their evaluation process is based on a methodology as outlined in the Information Technology Security Evaluation Manual (ITSEM) [3]. This document is still under development¹. In context with the complex Common Criteria (800 pages) major ambiguities have to be removed. The standards are of general characters and do not reflect the commercial interests. From a user and computer manufacturer perspective the standards have the following limitations:

- Endless disputes about the interpretation of the requirements.
- The interpretation process with respect to ambiguous technical terms is often not public².
- Ambiguity.
- Software maintenance versus re-evaluation.
- Different sets of criteria in different countries.
- Many standards have been specified under the control of governmental agencies. Some requirements, however, for governmental environments do not match with the requirements for the commercial market³.
- Some standards have a very high technical complexity and are not concise. The auditor and management acceptance is, therefore, very limited in a commercial environment.

Since the products of the computer manufacturers must be developed with respect to time and cost to market, the experienced evaluation process is often called the *criteria creep*.

In the meantime, ECMA, which was originally the European Computer Manufacturers Association and is now a worldwide association having members from Europe, the U.S., and Japan, has tried to fill the gap by providing a class of security functionalities, called the Commercial Oriented Functionality Class (COFC) [7], which reflect the commercial needs but at the same time allow independence of choice of assurance criteria and evaluation process. That means that the ECMA COFC can be used in conjunction with the CC or ISO standard, but could also be used with any other appropriate assurance scale or evaluation process. ECMA's objective was to reduce the technical complexity, keep the standard open for later extension to other environments and make it easy adaptable to any given constraints or requirements.

¹ A methodology for the specification of complex digital systems is missing in chapter 7. In addition, it is not obvious in which phase of the evaluation a specific tool or technique should be applied. Consistencies issues and system redesigns are also not addressed.

²The set of confirmed interpretations is not visible outside the NSA.

³Confidentiality, for example is less important for the financial market as for a governmental agency. In addition, repudiation aspects are important for the financial market, but not for a military environment.

The COFC is a 12 page document that presents a minimum set of security functions, satisfying the need of commercial companies who want to have their systems reasonable secure. Special emphasis was given to precise and unambiguous description of the security enforcing functions. Companies are asking what they need to be reasonably secure. Companies need a baseline standard to measure themselves against. They may also need a document to point to that indicates that they took reasonable and prudent precautions regarding the safety and security of their proprietary and personnel information.

3. ECMA's COFC Standard 205

3.1. General aspects

The ECMA COFC assumes a Corporate IT Security Policy of a commercial enterprise taking typical environmental and organizational constraint into account. As in reality, the Corporate Security Policy is based on a confidentiality policy, an integrity policy, an accountability policy and an availability policy. These dedicated policies are enforced by an appropriate IT security architecture that provides a specific set of security services and the associated security management. The security services and the security management are based on a specific set of protocols and mechanisms (security enforcing functions) which may be realized by non-cryptographic (access control) or by cryptographic means (symmetric methods, public key methods). Due to consistency and ease of operation a specific key management may be an integral part of the security management supporting specific security services and security mechanisms. With respect to the various system services which are applied, the security management system activates the adequate security enforcing functions. If cryptographic means are applied, the associated keys and parameters are protected such that unauthorized persons can't have access to it.

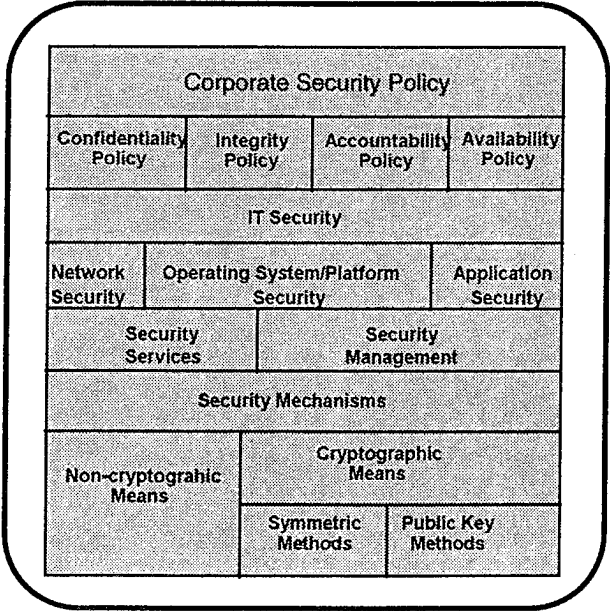


Figure 1: The different components of a Corporate Security Policy

Mechanisms and protocols as such are not specified in the standard with the exception of a password mechanisms for the case, that this mechanism is applied. A model addressing access control and accountability is shown in Annex A for a better understanding. Annex B gives a list of all used terms with definitions adopted from other documents and their references. The ECMA Standard 205 is limited to multi-user stand-alone systems, but open for extensions for example to interconnected systems.

3.1. The COFC standard

The objective of the Commercially Oriented Functionality Class (COFC) is the specification of widely accepted security functionality class for the commercial market. The standard addresses only IT security. Other security areas, like personal security, physical security, and procedural security are not covered. The standard defines a basic functionality class for the commercial market. It addresses multi-user, stand-alone IT systems and does not address networking or remote access. The standard is partitioned into several sections namely Identification and Authentication, Access Control, Accountability and Audit, Object Reuse, Accuracy, Reliability of Service, and Password specific requirements. On the basis of the imposed commercial market requirements and a risk analysis, the following security enforcing functions have been identified [7]:

<i>Threat</i>	<i>Security Enforcing Functions</i>
Outsider attack-Unauthorized access to the TOE.	Identification and Authentication prior to all other interactions.
Insider attack-Individual responsibility.	Unique Identification and Authentication, Accountability, and Audit.
Automatic logon attacks.	Number of logon trials.
Disclosure of authentication information.	Authentication information protection, Authentication information sharing, and Authentication information aging.
Disclosure of information.	Access Control, and Object Reuse.
Manipulation of information (accidental or intentional).	Access Control, and Accuracy.
TOE failure.	Recovery.
Natural disasters.	Data Backup.

It is beyond the scope of the paper to describe the COFC in detail. Instead some items are highlighted as being essential for a commercial environment and not addressed in other standards:

- Expiration of unused user identifiers.
- Disable users temporarily.
- Date of modification to objects.
- Application controlled access rights program path.
- Audit records to actions by authorized users.
- Survive of accountability control information at restarts of TOE.
- Alarm if unable to record audit trail.
- Dynamic Control for events recorded during normal operation.
- TOE software integrity.
- Data integrity.
- Status report of all customer specific security parameter.
- Data Backup.
- Default passwords.

For a commercial environment, these items are quite important, since a secure system and accounting management can only be realized if adequate means for backup, recovery, audit, integrity, and accuracy are supported. From our perspective, the dedicated mechanisms applied to implement these mechanisms may differ for the various systems. These means should, therefore, not be standardized.

With respect to the other ongoing activities in the standardization process, we have compared the COFC with other standards. Table 1 to Table 11 describe in detail the comparison with the CS1 and CS2 Protection Profile from the FC [5].

Table 1: Identification, Authentication and System Entry			
Functional Requirements	COFC	CS1	CS2
Unique Identification and Authentication	6.1.1	3.11	3.1.1
Identification and Authentication prior to all other interactions	6.1.2	3.11	3.11
Associate Information to users	6.1.3		3.1.4
Logon message	6.1.4		3.2.1
Display last access info			3.2.5
Number of logon trials	6.1.5		3.1.3
Number of logon sessions			3.2.2
Expiration of unused user IDs	6.1.6		
Session lock or terminate			3.2.6
Disable users temporarily	6.1.7		
User status information	6.1.8		3.1.4
Policy attributes control			3.1.4/3.2.4
Authentication information protection	6.1.9	3.1.3	3.1.3
Authentication information sharing	6.1.10		3.1.5.a
Authentication information aging	6.1.11		3.1.5.f
Protected mechanisms for Identification and Authentication		3.1.2	
No error feedback for Identification and Authentication			3.1.3
System entry denial by time			3.2.3/4
Password mechanisms	7.1		3.1.5

Table 2: Password Specific Requirements			
Functional Requirements	COFC	CS1	CS2
User-changeable, Password initialization	7.1.1.1		3.1.5.c
User-changeable, Password storing	7.1.1.2		3.1.5.b
Already associated password	6.1.10		3.1.5.a
Password aging	7.1.2		3.1.5.f
Expiration notification	7.1.3		3.1.5.g
Password reuse	7.1.4		3.1.5.h
Password complexity	7.1.5		3.1.5.i
Logging	7.1.6		
Default passwords	7.1.7		
Null password			3.1.5.d
Password generation algorithms			3.1.5.j
Password display suppression			3.1.5.c

Table 3: Access Control			
Functional Requirements	COFC	CS1	CS2
Authenticated user identification	6.2.1	3.4	3.7
Individual user	6.2.2	3.3.2	3.5.2
User groups	6.2.3	3.3.1/2	3.5.1/2a
Objects	6.2.4	3.3.3	3.5.3
Types of access rights	6.2.5	3.3.1	3.5.1
Default access rights	6.2.6	3.3.2	3.5.3.c
Precedence of access rights	6.2.7	3.5.3.a/b	

Table 3 cont.: Access Control			
Date of modification	6.2.8		
Verification of rights	6.2.9	3.3.3	3.5.3
Application controlled access rights	6.2.10		
Object reuse	6.5	3.3.4	3.5.4
Multiple access control policies			3.5.1
Change while active			3.5.2
List of no access to object			3.5.2
List of names of all groups	implicit in COFC		3.5.2.b
List of membership of any group	implicit in COFC		3.5.2.c

Table 4: Accountability and Audit			
Functional Requirements	COFC	CS1	CS2
Associate actions and users	6.3.1		3.4.2/3
Logging	6.3.2		3.4.2
Use of identification and authentication	6.3.2.1	3.2.2	3.4.2
Attempts to exercise access rights	6.3.2.2	3.2.3	3.4.3
Creation/deletion of object	6.3.2.3		3.4.4
Authorized user actions	6.3.2.4	3.2.2	3.4.2
Logged information	6.3.2.5	3.2.3	3.4.3
TOE restart	6.3.3		3.10
Copy audit trails	6.3.4		3.4.4
Alarm if unable to record	6.3.5		3.4.4
Select users	6.3.6	3.2.4	3.4.4
Dynamic control	6.3.7		3.4.4
Policy attributes		3.2.3	3.4.3
Tools	6.4.1		3.4.4
Select users	6.4.2	3.2.4	3.4.5
Automated copying/deletion			3.4.4
API			3.4.1

Table 5: TCB Protection and Ease-of-TCB-Use			
Functional Requirements	COFC	CS1	CS2
TCB address space		3.5.1	3.8.1
Noncircumventability		3.5.2	3.8.2
Default security parameters			3.12.1
Fail - safe			3.12..2
API			3.12.3

Table 6: Accuracy			
Functional Requirements	COFC	CS1	CS2
TOE software integrity	6.6.1	3.6	3.9
Data integrity	6.6.2		
API	6.6.3		

Table 7: Reliability of Service			
Functional Requirements	COFC	CS1	CS2
Recovery	6.7.1		3.10
Data Backup	6.7.2		3.10

Table 8: Object Reuse and Reference Mediation			
Functional Requirements	COFC	CS1	CS2
Object reuse	6.5	3.3.4	3.5.4
Reference mediation	6.2.9	3.4	3.7

Table 9: TCB Initialization, Recovery and Self Checking			
Functional Requirements	COFC	CS1	CS2
Data recovery	6.7.2		3.10.1
System re-start state	6.7.1		3.10.2
HW/SW operation check			3.9
Power-On test			3.9
Integrity test programs	6.6.1/2		3.9

Table 10: Privilege Association with TCB Modules and TCB Trusted Path			
Functional Requirements	COFC	CS1	CS2
Privilege Association			3.11
Trusted Path			3.3

Table 11: Security Management			
Functional Requirements	COFC	CS1	CS2
Maintenance mode mechanism			3.6.1
Modify policy parameters			3.6.2
Session inactivity			3.6.1.1
Logon/session time			3.6.1.2
Unsuccessful logons	6.1.5		3.6.1.3
Manage user registration	6.1.7		3.6.3
Listing security attributes	6.6.3		3.6.3
Routine control of system. resources			3.6.4
Object access control	Annex A		3.6.5

The following aspects have to be considered comparing the COFC with CS2:

1. COFC does not specify any assurance criteria. Therefore, the protection of the security management data (TCB) is not specified in COFC (Table 5).
2. One of the COFC main objectives is to specify a baseline standard which should be very easy to adopt in a commercial environment with respect to time and cost to market. Requirements, such as "Display last access info" (Table 1), are not supported by COFC since they have not been identified for the baseline set.
3. From our perspective, the following issues, not supported by CS2 are very important for a secure system in a commercial environment:
 - Accuracy (Table 6).
 - Audit records to actions by authorized users.

- Survive of accountability control information at restarts of TOE.
- Alarm if unable to record audit trail.
- Dynamic Control for events recorded during normal operation.

4. Future Work

The COFC has been adopted as an ECMA standard by the General Assembly in December 1993. Since January 1994, the responsible group TC36/TG1, is working on an enhanced COFC, called E-COFC, that addresses the minimal set of security requirements for interconnected single-user and multi-user systems.

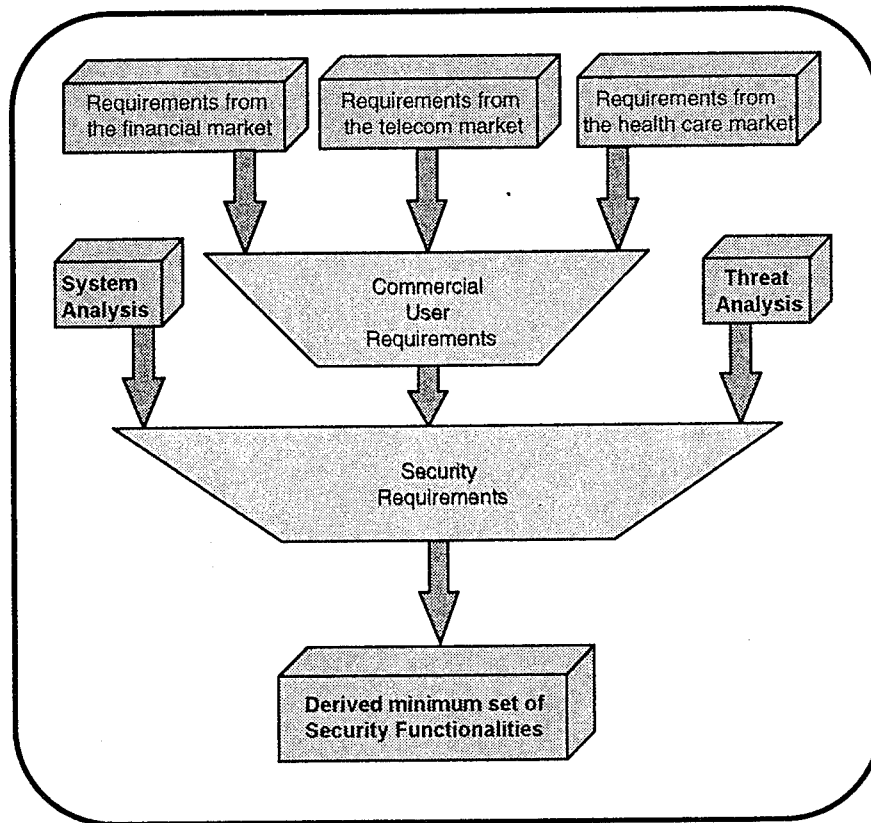


Figure 2: ECMA's approach for the specification of IT Security Evaluation Criteria

The E-COFC will be based on some of the security functions as defined in the COFC. The target for completion of a first draft version is January 1996. Presently, commercial requirements from different user groups in different countries are being investigated. Based on the results of this work, a threat analysis (communication, system, and user account compromise), and a technical system analysis (different network based operating systems), will be made to form the basis for the E-COFC and its validation (see Figure 2).

5. Conclusions

From our perspective, the COFC has the following advantages:

1. The standard is easy to understand since the specification of a minimum set of security functionalities reduces the technical complexity. The derived set provides a reasonable protection for commercial multi-user, stand-alone IT systems.
2. The standard is easy to adopt by different groups in a company, including the management level due to compactness (12 pages) and clear language.

3. The standard is concise. An auditor or a CEO can hand it to the internal security or IT management and ask simply, "Do we comply?"
4. The standard is based on today's commercial requirements avoiding military and governmental bias. Major efforts have to be undertaken to come up with a concise security evaluation standard that fulfills the constraints for these three very different environments. Today's operating commercial companies can't take the risk to wait such a long time.
5. The standard is independent of assurance criteria and evaluation methods. It is consistent with TCSEC, ITSEC and MSFR concepts. The COFC can, therefore, be applied with an appropriate assurance scale and methodology, which might be TCSEC, ITSEC, FC, CC or any scale of assurance criteria and evaluation methods provided by any other organization, if accepted by the user.
The COFC has been sent out as a contribution to the international standardization process.
7. The COFC is supported by computer manufactures from the U.S., from Japan, and from Europe. This international basis provides the commercial user with the necessary assurance that the mutual recognition of product evaluations is not limited.
8. The COFC provides the world-wide operating computer manufactures with a framework that is easy to adopt. They can, therefore, develop their products on the basis of this standard and fulfill the business requirements with respect to time and cost to market⁴.

ECMA will be promoting the COFC as a concise, easily understood standard that provides a necessary, but not sufficient measure of commercial IT security. That promotion will start as early as late this summer and will take place in Europe, in the U.S., and in Japan. In addition, the standard is submitted to different expert groups as contribution to the international standardization process. Within ECMA, TC36/TG1 is working on an enhanced COFC that addresses the minimum set of security requirements for interconnected single-user and multi-user systems.

References

- [1] "Trusted Computer Systems Evaluation Criteria", DoD 5200.28-STD, Department of Defense, United States of America, December 1985.
- [2] "Information Technology Security Evaluation Criteria (ITSEC) - Harmonized Criteria of France, Germany, the Netherlands, and the United Kingdom ", Version 1.2, June 1991.
- [3] "Information Technology Security Evaluation Manual (ITSEM)", Provisional Harmonized Methodology, European Commission, Directorate-General XIII, telecommunications, Information Market and Exploitation of Research, September 1993.
- [4] "The Canadian Trusted Computer Product Evaluation Criteria", Canadian System Security Center, Communications Security Establishment, Government of Canada, Version 3.0e, January 1993.
- [5] "Federal Criteria for Information Technology Security", Volume 1 and Volume 2, December 1992, National Institute Of Standards and Technology & National Security Agency.
- [6] "Common Criteria for Information Technology Security Evaluation", Version 0.9, CCEB-94/080- CCEB-94/085, 31.10.94.
- [7] "Standard ECMA-205, Commercially Oriented Functionality Class for Security Evaluation (COFC) ", ECMA, December 1993.

⁴Product evaluations based on the Orange Book need between 3 and 5 years.

Rating Network Components

Gloria Serrao
ATTN: C71
9800 Savage Road
Fort Meade, MD 20755-6000
(410) 859-4458
Serrao@DOCKMASTER.NCSC.MIL

The National Security Agency (NSA) Trusted Product Evaluation Program (TPEP) evaluates the security of computer products based on the Trusted Computer System Evaluation Criteria (TCSEC). As network security increases in importance, the Trusted Network Interpretation (TNI) of that criteria, and its guidance for partitioning networks into components should be revisited. This paper discusses the importance of a secure network architecture and design, the relationships between security policies, and analyzes what policies and assurances components must have to be rated as individual component types.

1 June 1995

Background

The Trusted Network Interpretation (TNI) allows for network products to be evaluated as a complete network system (Part 1 of the TNI) or as components (Appendix A of the TNI). Network components, according to the TNI, are products that do not support all the policies required by the Trusted Computer System Evaluation Criteria (TCSEC). Components are identified by the general type of policy that they do support: Mandatory Access Control (M), Discretionary Access Control (D), Identification and Authentication (I), and Audit (A). A component can provide any combination of M, D, I, or/and A functionality. There are 16 possible policies (combinations of M, D, I, and A). One of the 16 possibilities is that of no security policy (e.g., a network cable). This paper will address only individual M, D, I, and A components and what is required to give a component one of these ratings. In contrast to a stand-alone operating system which will contain all the functions necessary to accomplish MAC, DAC, I&A and Audit, a network component may share portions of the required functionality with other network components. For example, one component may be the Audit component for the network with the other components providing support for that Audit component by forwarding audit records to it for processing and review. This is a clear cut example, but what about the I&A or DAC component that itself does not store the database information it uses to determine correct identity (I&A) or correct access (DAC)? What portion of its functionality can a component require other components to provide? Can a component be given a D rating if it provides the "major" functionality required of a DAC component? This presents a difficult problem for evaluators who must decide whether or not the portions of the security policy enforced within a component are adequate for the desired rating. Evaluators must "step back" and view the component as a piece of an overall network system. If the component itself does not provide adequate security policy enforcement for the whole network, they must decide what items are lacking and how to sufficiently describe them in a Network Security Architecture and Design (NSAD) document.

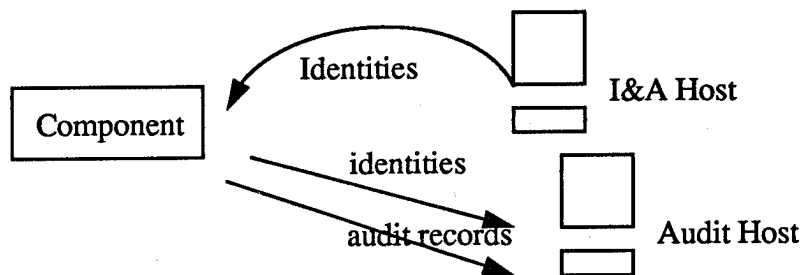
The Network Security Architecture and Design (NSAD)

The Network Security Architecture and Design (NSAD) document, as the title implies, addresses both an overall network security architecture (policies and objectives) and describes how the network should be built to comply with this architecture. The NSAD is a requirement stated in the Design Documentation section of the TNI. It must be provided for all network and network component products. It is described in Section 1.4.1 of the TNI as:

“The Network Security Architecture must address the security-relevant policies, objectives, and protocols. The Network Security Design specifies the interfaces and services that must be incorporated into the network so that it can be evaluated as a trusted entity. There may be multiple designs that conform to the same architecture but which are more or less incompatible and non-interoperable (except through the Interconnection Rules). Security related mechanisms that require cooperation among components are specified in the design in terms of their visible interfaces; mechanisms which have no visible interfaces are not specified in this document but are left as implementation decisions.” “The NSAD must be sufficiently complete, unambiguous, and free from obvious flaws to permit the construction or assembly of a trusted network based on the structure it specifies.” The NSAD must completely and unambiguously define the security functionality of components as well as the interfaces between or among components. The NSAD must be evaluated to determine that a network constructed to its specifications will, in fact, be trusted, that is, it will be evaluatable under these Interpretations.”

This short description does not adequately reflect the importance of an NSAD as a design document during the evaluation of a component. Since a component must be evaluated as a small piece of an overall network security policy, the overall policy must be defined as accurately as possible and should reflect a major investment of the vendor's time and resources. The NSAD sets down the foundational rules for a network, describes how the Network Trusted Computing Base (NTCB) is partitioned and how all the trusted system requirements are satisfied. The NSAD must describe a clear division between what security functionality is enforced in each NTCB partition; including the partition (component) under evaluation. It must discuss what security-related cooperation is needed between components. It must tell what is expected of anything connected to the component and include known items that could undermine the component's security policy. It must describe in detail what things must be present for the component's security policy to work properly. For example, if a component provides C2+ (equivalent to B3) Discretionary Access Control (DAC), the NSAD must describe a network level B3 DAC policy to include the ability to deny access based on “a list of named individuals AND a list of groups of named individuals”. Any network or component that is evaluated must not only meet the TCSEC requirements and TNI interpretation but must also conform to the NSAD. Conformance to the NSAD ensures that all the pieces of the network can be combined and result in a network that enforces the overall network security policy.

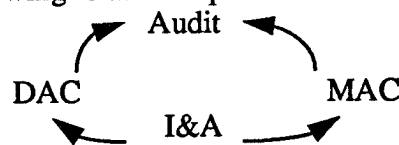
Evaluators and vendors of network components must be careful when using an NSAD to describe what is required of other components. As an NSAD broadens in scope, the component under evaluation becomes less functionally complete. For example: if a network component itself does not provide audit records of administrator actions and does not include a way to identify the individual taking the action, could it rely on one host to provide the I&A information and yet another host to correlate the audit records with the individual identities? Can this reliance on two different hosts to meet a requirement be documented in an NSAD so that the component meets the audit requirement?



No, while it is true that these reliances must be documented in an NSAD, the component would not meet the audit criteria merely based on its assumptions regarding the two hosts. The audit criteria requires that an administrator be able to selectively audit actions based on user identities and this could not happen at the component under evaluation.

Security Policy Relationships

It is important to understand the inter-relationships of the main security policies within a generic network system. Although components are rated based on the general policy elements the component supports, the following relationships should be considered when analyzing a component:



Information supplied during identification and authentication of a user is used by the Trusted Computing Base (TCB) to make Mandatory Access Control (MAC) and Discretionary Access Control (DAC) decisions. Audit utilizes the I&A information combined with MAC and DAC access checks to formulate audit records. When the TCB is provided a user name and password, it compares it to the correct password and allowed classification level. If correct, the TCB associates this identification information with all actions taken on behalf of a user. Given this inter-relationship diagram, components that do not themselves meet the requirements for a given security policy (M, D, I or A) must still provide necessary information to other components in the network so that all policies can be met in the overall network architecture. The information that a component provides to other components must be captured in the NSAD and should also be described in the Trusted Facility Manual (TFM) for the product.

How TCSEC Requirements are Reflected in Component Ratings

When the TNI addresses components individually (in Appendix A, Section 3), only MAC components can meet levels of trust higher than C2. DAC and Audit are eligible to be C2+ components provided they meet the requirements for B3 DAC and Audit. Therefore, a C2 product could contain C2+ DAC and Audit; but because the assurance requirements that accompany MAC are not present, only a C2+ can be achieved. The C2+ designation indicates that although the features meet requirements above C2, the features are not accompanied by B3 assurances such as system architecture and penetration testing. This is easy to understand with a DAC component which may indeed have high assurance but because DAC is inherently flawed, nothing has been accomplished. B3 DAC is an oxymoron because access controls are based on user discretion and can be propagated unwilling or unknowingly. For example, User A may create a file and grant User C no access. User B, however, has read access and can read the file, copy it, and make it accessible to User C.

The TCSEC I&A requirements do change at B1. However, the change is MAC related and states that the I&A data must be used to ensure that the sensitivity level and authorizations are properly associated with the user identification. When the TNI was written and the component requirements revisited, this portion of the TCSEC I&A requirement was moved to the component MAC requirements.

In practice, when components are evaluated as any combination that includes MAC (i.e., MA, MI, DA, etc.) the MAC policy is examined first. The maximum level of trust that can be given to a component is that which imposes the highest requirement. MAC requirements encompass assurance requirements which, in turn, apply to the I&A, DAC and Audit functions of the component. Therefore, whatever rating the MAC portion receives is the resulting rating for the component. The composition guidelines in Appendix A, Section A.2 "Composition Rules" of the TNI gives more detailed guidance for combining component ratings. For example, when evaluators analyze a B3 MDI component, it will contain B3 assurances and will thus be able to be called a B3 MDI component provided the D and I functionality meet the B3 requirements.

Analysis of Individual Component Ratings

Keeping in mind the importance of the NSAD and policy inter-relationships, what is required of each individual component type at the various levels of trust? To analyze this and to point out questions for the evaluator community, I have reviewed the applicable sections of TNI Part 1, "Interpretations of the Trusted Computer System Evaluation Criteria", and TNI Appendix A, "The Evaluation of Network Components". This criteria was reviewed specifically with regard to past evaluation community discussion about whether or not a component needs to contain all elements of the particular security policy (MAC, DAC, I&A, or Audit) in order to be rated for that individual policy.

An early network component evaluation (an MI component) did not include a network management workstation and, therefore, contained no storage databases for MAC and I&A information. However, the MAC and I&A criteria call for storage of security levels and identities. This evaluation proceeded based on a technical decision that stated "because of the distributed nature of networks and their functionality, network components can rely on other components to provided pieces of their functionality. This decision broke component security mechanisms (D,I, M and A) down into subfunctions which are defined below. In order to meet the appropriate criteria, components did not have to be solely responsible for meeting the entire security policy but may perform only the appropriate subfunctions as defined and summarized below. Note that all components must "enforce" audit; i.e., produce audit data for their own auditable actions.

Decision: the check being made and the logic of the algorithm used in the check.

Example: I&A decision compares password entered with correct user password

Enforcement: confidence that functionality will always be performed

Example: MAC enforcement ensures access for the subject to the object; no other object, no other access

Audit Enforcement: ensures an audit record is generated whenever an auditable event occurs

Storage: Storing the information used in (or resulting from) the check.

Example: for MAC, mapping of objects to labels and of subjects to labels

for I&A, mapping of ids to authentication data.

Table 1: Network Component Subfunctions

Type of NW Component	Subfunction Required
I&A	Decision, Enforcement, Audit Enforcement
MAC	Decision, Enforcement, Audit Enforcement
DAC	Decision, Enforcement, Audit Enforcement
Audit	Decision, Enforcement, and Storage

Note that storage is not required for MAC, DAC or I&A components. This particular point, as well as others, will be examined for each type of component.

I&A Components

TNI Part 1 (Network) Interpretation:

Individual accountability for security relevant actions is still the objective for I&A. However, the TNI allows this accountability to be met by host identification instead of by I&A of an individual user provided that the host identifier implies a list of specific users. This would allow I&A to be performed by a host IP address for which the host contains a list of individual user names. I&A information can be passed from one component to another. Each component does not need to reauthenticate the I&A data provided the information is protected during transit.

TNI, Appendix A , Section A.3.4 (Component) Interpretation:

- I&A mechanisms can only meet C2 requirements and include the following requirements that are applied without further interpretation: I&A, Object Reuse, Security Features User's Guide, Security Testing, System Architecture, System Integrity and Test Documentation.
- I&A mechanisms do not include mechanisms to completely support any of the three other network policies.
- Audit is specifically mentioned "the I-component shall produce audit data about any auditable actions performed by the I-component"
- Because an I&A component does not maintain audit files or provide mechanisms for examining them, only the mechanisms for exporting audit data must be defined in the Trusted Facility Manual (TFM).
- The design documentation for an I&A component must contain a description of the protocol used by the I-component to export authenticated subject identifiers to other components

Analysis:

A key phrase in Appendix A, Section 3 is "do not include mechanisms to completely support any of the three other network policies. By using the word "completely" it is clear that while the I&A component does not have to perform DAC, MAC and Audit (it alone is not responsible for the completion of these policies) but it must be able to supply the required information to other network components and state how this is accomplished in the NSAD and the Trusted Facility Manual (TFM).

The use of host addresses for I&A actually allows the authentication of a users identity to take place on the host. Thus, the host must be a trusted host and the requirement for a trusted host would need to be stated in the TFM. Based on the identifier passed to the network component from the host, would the network component authenticate the identifier? If not, it is not appropriate to assign the

component an I&A rating because instead of performing I&A, the component is merely using an identifier passed from another component in order to perform a MAC or DAC access check. In many cases, a component will perform some type of authentication. For example, based on the identifier provided, is this host a valid host? If the component does this, it is eligible for an I&A rating. A component can also be an I&A component based on the I&A it performs for a console operator or system administrator, provided the authentication data is protected and individual accountability and auditing can be accomplished.

If storage is not required for a component to be an I&A component, the password file or other mapping of identities to authentication data can be stored on another component. The other component would then be trusted to supply a correct mapping. Part 1 of the TNI, Section 2.2.2.1 Identification and Authentication states:

“Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user’s identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user.”

The requirement for protection of authentication data implies storage of that data. In addition, how can an I&A component be tested without its authentication data? If an I&A component is not required to store its authentication data, one could envision a network that has an I&A component but cannot perform DAC and MAC checks. If a component is rated as an I&A component, shouldn’t an integrator expect it to perform I&A for his network without reliance on another trusted component to store the database?

The “decision” that an I&A component performs is the comparison of the authentication data entered with authentication data maintained. If it does not maintain the authentication itself can it adequately “decide”?

Remembering that the I&A requirement is redefined for components and does not include the association of an id with a clearance level, an I&A component does not need to do this association, however, a M-component would.

Mandatory Access Control (MAC) Component

TNI Part 1 (Network) Interpretation:

A MAC policy and sensitivity labels must be present in the component. MAC must be exercised over all subjects and objects under its control. For a subject in one component to access an object in another component, there must be the creation of a surrogate subject in the second component which acts on behalf of the first subject. The label requirements are significant and address how a component NTCB partition must assign a label to non-labeled data it receives. The TNI expands the label requirement to address integrity of labels. The requirement for exportation of labeled information states that sensitivity labels remain correctly associated with exported information. The accurate representation of sensitivity labels throughout the network system must be described in the network security policy. The MAC requirements span from B1 to A1. At B2, access controls extend to “subjects external to the TCB and all objects directly or indirectly accessible by these subjects”. B2 requires device labels and subject sensitivity labels, as well as trusted path.

TNI, Appendix A , Section A.3.1 (Component) Interpretation:

- M-components can meet B1, B2, B3 and A1 requirements and are rated according to the highest level for which all the requirements of a given class are met.
- The M-component shall produce audit data about any auditable actions performed by the M-component and provide a mechanism for making the audit data available to an audit component.
- Requirements that apply to M components without further interpretation are: Configuration Management (for B2 and above), Design Documentation, Label requirements (including device labels, exportation, human-readable output, and integrity), MAC, Object Reuse, Security Features User's Guide, System Integrity, Test Documentation, Trusted Distribution (A1), Trusted Facility Management (B2 and above), Trusted Recovery (B3, A1).
- If a component does not support direct terminal input there is no Subject Sensitivity Label requirement. Likewise, if an M-component does not support direct user input, there is no Trusted Path requirement.
- Interpretation of the B2 and above requirement to define the user interface to the TCB is that user interface means the interface between the reference monitor of the M-component and the subjects external to the reference monitor shall be completely defined.
- To support Covert Channel Analysis, the M-component must provide a mechanism for making audit data for any necessary covert channel audits available to other components.
- The mechanisms and protocols used to export audit data have to be provided in lieu of describing how to examine and maintain audit files.
- The I&A requirements state that the TCSEC I&A requirements for establishing a user clearance by mapping user ids to labels are reflected in M components. Thus M-components are responsible for using I&A data to map user identities to labels.

Analysis:

Again, a MAC component need not provide complete support for the three other policies, but is expected to provide appropriate support based on the NSAD. A MAC component can play a major role in building a network from single level Local Area Networks (LANs) or from one or more multiple level networks and one or more single level networks.

The MAC policy for a network must address how labels are transferred from one component to another and what is required from each component for the association of the labels with appropriate objects.

The Label requirement in Part I of the TNI points out that labels may include both secrecy and integrity components. Thus far, TPEP evaluations of network products have not included evaluating a mandatory integrity policy. Although evaluators may state that an integrity feature is present (e.g., checksums), they do not evaluate the feature with regard to strength or appropriateness. If a sponsor proposed an NSAD with a mandatory integrity policy, evaluators would have no accompanying requirements for evaluating the integrity policy since integrity is not a separate requirement under the TCSEC. The use of cryptography seems an obvious solution for label integrity. Products in past TPEP evaluations have typically met the label integrity requirement by storing labels in a specially protected file and requiring that they be physically protected during transit over a network wire. The use of cryptographic checksums on labels is also a viable solution.

The "decision" of a MAC component requires the mapping of a user id to a clearance level since this requirement was moved from the I&A component requirements. Based on this added requirement, a MAC component must store and maintain the I&A data that uses to map ids to labels.

A component that simply performs access checks based on subject and object labels is not a MAC component because correct MAC relies on the correct association of ids to sensitivity labels. Although it might be logical to state that another trusted component (an MI component perhaps) can perform the I&A and associate identifiers with clearance levels, the TNI specifically states that to be rated as an M component, the I&A mapping must take place in the M-component itself.

The audit requirements must be kept in mind even while designing and/or evaluating an M-component. M-components are required to "produce audit data about any auditable actions performed". Therefore, the required actions and the required formats for auditing must be in place for an M-component. A cursory reading of the TNI could lead to the conclusion that an M-component need not audit those events specifically enumerated in the Audit requirement (since it is not an Audit component). The MAC requirement in Part I of the TNI does not state that all security relevant events must be audited, however the Appendix A statement above and its use of the words "auditable actions" logically means that all security relevant actions performed by the M-component must be audited. In order to fit into an overall secure network, auditing of the M-components internal security related actions is mandatory. The NSAD for the network should be reviewed to ensure that the M-component audits are appropriately capturing security related events. The M-component protocol for correlating and/or exporting audit records must be stated in the NSAD and TFM.

Protocols become of major importance in a MAC component. The sponsor of a component under evaluation must specify how labels are done (e.g., in accordance with RFC 1108). In addition, the component NSAD must address the common representation of security levels for the overall network.

MAC controls must be enforced at the interface of the reference monitor for each NTCB partition. MAC mechanisms, in contrast to DAC mechanisms in a network system, must be enforced equally for the entire system. For example, a label assigned to file A will be associated with that file when it is accessed anywhere on the network.

What about storage within a MAC component? Does a MAC component need to store the database it uses for mapping objects to labels and subjects to labels? In Section 3.1.1.4 of the TNI, the Interpretation section states "In a network, the responsibility of an NTCB partition encompasses all mandatory access control functions in its component that would be required of a TCB in a stand-alone system." This statement implies that the labeling information should be present in a MAC component. How can you test an M-component without access to the data it uses for performing security relevant decisions?

Discretionary Access Control (DAC) Component

TNI Part 1 (Network) Interpretation:

DAC mechanism(s) may be distributed over the partitioned NTCB. Network components that contain only internal subjects (subjects that do not directly act on behalf of users) might not contain DAC. Network identifiers (e.g., internet addressees) can be used as group identifiers as long as specific individuals are implied by the group identifier. The DAC mechanism can be implemented at the interface of the reference monitor or in subjects that are a part of the NTCB in the same or different component.

TNI, Appendix A , Section A.3.2 (Component) Interpretation:

- D-components do not necessarily include the mechanisms to completely support MAC, I&A or Audit
- Can be rated C2 or C2+
- The D-component "shall produce audit data about any auditable actions performed by the D-component" and make the audit data available to an audit collection component.
- Requirements that can be applied without further interpretation are: DAC, Object Reuse, Security Features User's Guide, Security Testing, System Architecture, System Integrity and Test Documentation.
- For design documentation, a component must meet the requirement as stated and include a description of the protocol used to communicate user identities with other components. This further interpretation is required because a DAC component does not maintain user I&A information but in most cases must use some form of the I&A data for making DAC decisions.

Analysis:

The TNI allows for the DAC mechanism(s) to be distributed over the partitioned NTCB. This means that a DAC mechanism can exist for files and directories of a host and also for objects such as data packets on a router. DAC is enforced locally. Because of the distributed nature of DAC in a network, a broad DAC policy must be described in the NSAD.

Although it is feasible to use a DAC component in a secure network, it relies on I&A data in order to perform access checks. For DAC, I&A can be provided by another component and the design documentation for the D-component must describe the protocol used to receive I&A information from another network component.

DAC can be applied/changed by users. For example, an access control list (acl) assigned to a file may be different when viewed from different systems on the network.

The DAC requirement states that "the enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals or defined groups of individuals, or both, and shall provide controls to limit propagation of access rights." This statement implies that the enforcement mechanism is a part of the DAC component and would be defined by the security policy of the DAC component.

Audit Components

It is difficult to define what an audit component is fully responsible for providing without reviewing the TCSEC which states that an audit component must create, maintain and protect an audit trail of accesses to the objects it protects. It must record the use of I&A mechanisms, the introduction of objects into a user's address space, deletion of objects, actions taken by computer operators and administrators and other security related events. An audit component must provide specific information (date and time, user, etc.) as a part of the audit record. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity and/or object sensitivity level (sensitivity level applies to B1).

TNI Part 1 (Network) Interpretation:

Auditing in a network may extend to TNI Part II Security Services and may include items specific to networks (i.e., when a component goes down and subsequently rejoins the network).

It is an important function of auditing within a network to include all identification information with audit trails so that they can be properly correlated from different hosts. The term "users address space" is extended for object introduction and deletion events to include address spaces being employed on behalf of a remote users (or host). Audit information must be stored in machine-readable form. Of special note is the phrase in the Interpretation section which states "Furthermore, a component of the network system may provide the required audit capability (e.g., storage, retrieval, reduction, analysis) for other components....." If this statement is taken as a definition of the required audit capability, in order to get an A rating, a component must store the audit records, be able to retrieve them, reduce them and provide for the analysis of the records. Following that the I&A and DAC requirements allow host addresses as identifiers, audit on a network need not be for a specific user provided the host can identify the individuals represented by the group address identifier.

TNI, Appendix A , Section A.3.5 (Component) Interpretation:

- Provide network support of the Audit Policy (per TCSEC)
- Do not include the mechanisms to completely support MAC, DAC and I&A
- Are rated as either C2 or C2+
- Audit component requirements that can be applied without further interpretation are: Audit, Object Reuse, Security Features User's Guide, Security Testing, System Architecture, System Integrity, Test Documentation and Trusted Facility Manual
- The design documentation for the A component must include a description of the protocol used by the A component to import Audit data from other nodes.

Analysis:

Audit is a difficult requirement for a network component to meet unless its chief purpose is to collect, store, and provide audit records for review. The audit requirements lead to the conclusion that an Audit component itself must 1) create, maintain and protect audit records; 2) allow the system administrator to choose which actions and what users are to be audited and 3) provide the storage, retrieval, reduction and analysis functions for audit of the overall network. If any one of these various functions are missing within a component, it cannot receive an A rating. For example, if a component performs auditing, cuts audit records (and/or receives them from other components), correlates audit records, protects the records while they are local, and allows for the selection of what can be audited but does not store the audit records itself, it may not be considered an A component. This can lead to a further definition of "storage". Does temporary storage count, for example, if the component temporarily caches the audit data until it is uploaded to its permanent storage component? Another example of an incomplete audit component, is one that performs audit, cuts audit records (and/or receives them from other components), correlates audit records, stores and protects the records but does not allow for the selection of what can be audited. In other words, this component must utilize a separate pre- or post-processing tool. Strictly following the criteria, this component would not be an Audit component because it does not allow "the ADP system administrator to selectively audit the actions of any one or more users based on individual identity or object security level." However, there is a TCSEC interpretation (C1-CI-02-85) which states that "Audit reduction tools, when supplied by the vendor, must be maintained under the same configuration control system as the remainder of the system." This would allow an A component that merely maintains a pre or post-processing tool under configuration management to be rated as an Audit component without requiring in-depth analysis of the tool.

The words in Appendix A, Section A.3.5 "Audit only components are components which provide network support of the audit policy..." seem to contradict other audit sections of the TNI and

the way in which the Audit criteria has in the past been levied on components. The words "network support" are weak and could be construed to be a MAC component that produces audit records. Actually, all of the audit functionality of a network is contained in an Audit component. Section A.3.5.4 which discusses a representative application of A-components says that "The A-component provides auditing functions for the network as a whole" and this is the more common and correct definition.

Also in Appendix A, Section A.3.5.4, entitled "Representative Application of A Components", there is a scenario that "the Auditor may access the A-component via another component, in which case the A-component would be responsible for enforcing an access control policy that defined which users (i.e., the auditor) could view the audit data." It is unclear whether "the Auditor may access" means just a review of the audit records or whether the Auditor could select what to audit via another component (e.g. terminal). It could be acceptable to simply review audit records from another terminal, provided the Audit component enforced the required access control policy, but if that other terminal (and its interface to the system administrator) is also the means to allow the selection of what is to be audited, the terminal and application used would need to be placed under configuration control, as stated in Interpretation C1-CI-02-85. This is required because the selection of what to audit cannot be performed at an untrusted interface.

Audit records must be supplied to the Audit component from a component of a level of trust at least as high as the Audit component.

Although the TNI requires that A components store their audit records, stepping back from the TNI for a moment, one could easily envision the storage of audit records on another component provided guidance is given with regard to the interface, what occurs when the storage component goes down, and how many audit records could be lost. The design documentation for the Audit component would need to describe the protocol for receiving audit records from the storage component. Testing could still be a problem, but a test tool that emulates storage is feasible. In contrast to I&A, MAC and DAC components who use their databases to implement immediate access control decisions, audit information is used to analyze previous events.

Conclusions

1. The Network Security Architecture and Design document is the foundation for a secure network. Components must conform to an NSAD and the NSAD should be evaluated to determine that not only does the one component under evaluation fit into the architecture but to also decide if there are other likely candidates for component evaluations within this architecture.

2. The inter-relationships between the I&A functions and the MAC and DAC functions must be observed closely for components that are being evaluated as a M or D component. Evaluators must question how a M or D component is being provided user identifiers. In addition, the interrelationship between a M or D component and an A component must be examined in detail and include protocols for transporting audit records.

3. For each type of network component, there are requirements of the TCSEC and TNI that logically result in the storage and maintenance of data used by that component for security policy enforcement. Therefore a component should not be given a rating for a security functionality (MAC, DAC, I&A or Audit) for which it does not meet all the requirements. This includes the storage and protection of data utilized. Specifically, with regard to databases containing information used in access decisions, they should be considered a part of the component Trusted Computing Base (TCB)

and must be protected and controlled as such.

4. Although network functionality is distributed over a network, the TNI does not provide a way to dissect the requirements levied on a M, D, I or A component. Appendix B of the TNI entitled "Rationale Behind NTCB Partitions" states that there should be a clean decomposition of the overall network security policy into policies for the individual components.

5. Because of the policy dependencies between all types of components, individual M, D, I or A components are of limited value. Components that provide some combination of security functionality are more valuable.

Recommendations

1. The Trusted Product Evaluation Program (TPEP) should assess the value of components on the EPL and under evaluation to date. The TPEP should question whether or not they provide a useful preliminary step for the eventual evaluation of a network. If they do serve as a preliminary step, TPEP or some other organization should encourage vendors to develop other products to complete the secure network.

2. NSADs should be used as building blocks in much the same way as components. NSADs are sometimes proprietary and this limits their use by other vendors who may be able to design a component conforming to the NSAD. NSADs could be generalized into targeted secure architectures, components presented for evaluation to a targeted secure architecture could be evaluated at a more rapid pace than components with a "new" NSAD.

3. Guidance should be provided to vendors for writing a comprehensive NSAD. Because this document is important to a component evaluation, it should be reviewed early in an evaluation. Under the current process, an NSAD should be reviewed before the Intensive Preliminary Technical Review (IPTR).

4. As new network component products are proposed for TPEP evaluations, the technical review should center on the proposed secure network architecture. Is it realistic? Does it really provide a secure network? These questions should be asked together with questions about the technical viability of the component. The market review should determine whether customers' needs are met by this product and whether the market for the product supports the allocation of evaluator and vendor resources for the evaluation.

5. Those tasked with reviewing and refining the Common Criteria should consider how secure network architectures (NSADs) and components will be addressed. They should use the advent of a new criteria to address any TNI inconsistencies. Whether or not the TNI is too constraining for today's type of network products should also be reviewed.

References

TCSEC DOD 5200.28-STD, Department of Defense Trusted Computer System Evaluation Criteria, dated December 1985.

TNI NSCS-TG-005, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, Version 1, July 1987.

The Interpreted Trusted Computer System Evaluation Criteria Requirements, dated 12 January 1995.

ANALYSIS REQUIREMENTS FOR LOW ASSURANCE EVALUATIONS

James L. Arnold Jr.
Attn: C41
9800 Savage Rd.
Ft. Meade, MD 20755-6000

A study was conducted by TCSEC product evaluators of the analysis-related objectives, requirements, and evaluation processes for TCSEC class C2 and B1 product evaluations. The level of analysis conducted during these evaluations has changed over time. This report begins to identify a level of analysis that may be more appropriate and concludes with a set of recommendations for the future of analysis in C2 and B1 product evaluations.

1. Observation

A criticism of C2 and B1 product evaluations, as they are performed by the Trusted Product Evaluation Program (TPEP), is that they take too long. This, in turn, is often attributed an analytically overzealous evaluation community.

1.1. Inconsistent and changing level of analysis

Throughout the history of the TPEP, the level of analysis performed across the set of products under evaluation has been inconsistent and generally increasing over time. There are a number of reasons for this, not the least of which being the simple fact that "level of analysis" has not been formally defined in quantifiable and useful terms. Without such a definition, some of the principle characteristics of analysis, specifically breadth and depth, have been allowed to change with the advance in state-of-the-art in trust technology and with the variances in product design and evaluation team expertise and experience.

The growth in breadth of analysis has been realized in that each problem that has been identified by evaluators is looked for in every subsequent product evaluation. The result has been an ever growing list of problems which tend to broaden each subsequent analysis. Note that there is no well defined list maintained or known by the TPEP. This so called list is embodied in community knowledge which exists in a number of forms (e.g., individual minds, forum transactions, briefing slides, conference proceedings, CERT advisories, and interpretations). The informal nature of this list is a primary contributor to the inconsistency of its application.

The growth in depth of analysis has been realized more simply from the fact that many evaluation teams do not know what is important in arguing that the relevant requirements are met. For example, while it is clear that implementation details are necessary in understanding a class A1 product, it is not so clear in a C2 product. Evidence that a C2 product meets the requirements is supposed to be based upon design documentation and testing, but what if the documentation does not exist or what if the mechanism cannot be tested? Many evaluation teams spend energy analyzing things outside the Trusted Computing Base (TCB). This is because either they do not under-

stand what is or should be in the TCB or they feel it necessary to perform analysis trying to make those determinations. Evaluation teams also, in the lower evaluation classes, may not know what constitutes an acceptable argument. It is not sufficient simply to assert that something meets the requirements, and it is generally too much to provide implementation details. The currently unanswered question is where the "sufficient" line is drawn between these two extremes. Once this line is known, vendors would need provide no further detail, and evaluation teams would not be expected to look for further detail. Note that absence of documentation has historically resulted in looking at too much detail. In the absence of a sufficient argument (e.g., provided by the vendor in the design documentation), the team has had to build that argument themselves from less abstract (e.g., implementation) details. A point hidden in this reasoning is that it is not the level of detail that is necessarily key, but rather the questions that must be answered or arguments that must be made as a result of analysis.

1.2. Evaluation team performance

As trust technology has become widely known, there has evolved pressure to perform evaluations faster, if not better. This is not to say that there has not always been pressure for expediency. Vendors have, in fact, always wanted faster evaluations, primarily because faster generally means that a more current product can be brought to market, perhaps even at less cost to the vendor. Customers have also wanted faster evaluations, this desire being related closely to an understanding of trust technology and a growing perceived need for trusted products. However, these pressures have resulted in only minor improvements in the TPEP to date. More recently, this pressure has grown and additional pressure has developed from direct competition in the form of evaluations performed by other nations and the TPEP's own internal desire to provide better (e.g., more timely and more useful) products to their customers. In order to produce a more timely product without impacting overall assurance and without consuming more resources (e.g., overtime or fewer concurrent evaluations), guidance must be produced that is capable of helping an evaluation team do what is necessary and nothing more. Such guidance would also serve to help a vendor understand exactly what need be produced and provided for an evaluation to succeed as well as to guide the quality assurance checks (e.g., Technical Review Board (TRB)) in defining what should be expected of an evaluation team.

1.3. "Common ground" understanding

While a well defined set of evaluation guidance would help vendors and evaluators understand exactly what is necessary to successfully complete an expedient evaluation, it would also serve as a very useful tool for users. Such guidance would provide users a better understanding of what they are getting and the risks involved. This is especially important if the "level of analysis" for future products is substantively changed from that of the past, assuming that users generally understand what they are currently getting.

2. History

The changes in the level of analysis performed by evaluation teams for the TPEP have been very gradual. Though attempts have been made to ensure consistency, they have been effective only in

preventing sudden and obvious change.

2.1. Evaluator differences

The level of analysis problem has existed in the TPEP since its beginning when its evaluators were trying to figure out what to do. Evaluators, including TRB members, have been doing different things that each perceives as the right thing based upon personal experiences and biases. These differences have been allowed to gradually change the process, unchecked, largely because of a lack of recognition of this evolution and any subsequent action (e.g, acceptance, correction, direction).

There has been little done to prevent TRB members from gradually expanding the scope of analysis, primarily by asking more and more questions. There has also been little done to prevent teams from doing too much work. In fact, there has been incentive to do more. For example, a team that suffers a bad TRB experience will likely over-prepare in a hope that the next experience would not be so harrowing. Even though TRB recommendations often include items to reduce evidence presented about non-security relevant or otherwise uninteresting details, those recommendations are seldom enforced.

2.2. TRB and consistency

A very important TRB function is to ensure consistency, but it has been impossible to prevent the expectations of that group from growing over time. A reason that evaluations have a reasonable amount of consistency at any particular point in time is because the TRB group is small enough to maintain a common expectation with the rate of growth in that expectation set being very slow. Nonetheless the expectations are not easily captured in a form that an evaluation team can use to ensure success without over-analysis.

2.3. Chief evaluator/senior evaluators and consistency

There have been a number of roles defined in the TPEP to help ensure consistency. The senior evaluator roles were designed to allow a small set of individuals to monitor and interact with evaluation teams and to provide more immediate (as opposed to TRB milestones) guidance relating to issues that need be resolved and analysis that need be performed. This function was never fully realized largely because the assigned individuals lacked appropriate guidance and understanding themselves and because this function was not properly prioritized to provide the necessary resources.

The role of the chief evaluator has been more directly related to TRB oversight. It was quickly realized that one individual could not provide adequate oversight for all evaluations, hence the aforementioned introduction of senior evaluators. The chief evaluator could, however, monitor all TRB activities to help ensure that the TRBs were being consistent and that their recommendations actually reflect current evaluation expectations. Unfortunately, even this role has been unable to halt the advance of expectations.

2.4. Time constraints on potential solutions

Since the TRB has the best understanding of what it expects during TRB milestones, it is also

most capable of commenting on each team's analysis relative to those expectations. As noted earlier, TRBs often recommend that certain analyses presented are non-security relevant or are uninteresting and should therefore not have been performed and should not be included in the Final Evaluation Report (FER). However, such comments are often disregarded simply because of the level of effort that would be required to remedy the situation (i.e., determine and document exactly the right things). The ideal situation would obviously be that this never happened, but from a resource perspective it has generally been better to leave it alone rather than to fix it after the fact. Due to the push to get each product evaluated quickly and a corresponding limitation of resources (both on behalf of the evaluation team and the TRB members) to examine what might have been wrong, evaluation teams have not learned from these occurrences.

2.5. Product evolution

It should be noted that part of the cause of the evolution of expectations has been due to the types of products being evaluated. When the TCSEC was written, the authors apparently envisioned relatively simple, stand-alone, monolithic type systems that would be locked away in a lab and accessed via remote terminals. Those types of systems are no longer the rule, but rather have become the exception. The systems of today are very complex, and the complexity is still growing. They include network interfaces, non-uniform and complex TCB interfaces, redundant object types, multiple processors, intelligent devices, graphical user interfaces, desktop and laptop architectures, etc. With the growth of these types of products, it has seemed reasonable to the TPEP to expand both breadth and depth of analysis to cover the new issues.

3. Intended use of products

Since some of the breadth of analyses performed in the TPEP today are obviously incorrect (e.g., non-security relevant parts), it is reasonable to assume that the depth of analysis may also be incorrect. When evaluators and TRB members are asked why they perform or expect the analyses that they do, they give responses ranging from "it's what everyone else is doing" to "it's the right thing to do." The former is obviously unjustifiable, while the latter could have some merit. What exactly is the right thing to do? The right thing should be based on the intent of the criteria and the needs of the anticipated customers. The intent of the criteria has been largely characterized in the text of the TCSEC, but the needs of the customers vary and are often confused with desires, resulting in unreasonable expectations. For example, many customers are using B1 products where B2 products should be used. In this and similar situations, customers expect more from those products than the TPEP has determined (by evaluation) that the vendor provides.

3.1. Intended product environments

The intended uses and environments for C2 and B1 class products can, at least to a point, be extracted from the text of the TCSEC and related guidelines. This information, while contained in the requirements themselves to a small extent, is actually better provided in the text surrounding the requirements (e.g., the objectives of the classes and the guidelines in the TCSEC appendices).

The TCSEC section "Structure of the Criteria" [TCSEC85, p.5] leads one to believe that there is very little difference in assurance between C2 and B1 (division C and B1 are grouped in relation

to assurance) and that the more substantial difference is in function ("each division represents an improvement in... protection of sensitive information"). These also explicitly state that the primary means of assurance for both C2 and B1 is testing. This, coupled with minor differences in the C2 and B1 requirements, implies that a similar level of analysis should occur at C2 and B1.

The various class and division objectives stated in the TCSEC [TCSEC85, pp.12,15,19,20,26] further clarify the intended use of division C, specifically C2, products and the intended differences between C2 and B1. The class C1 and C2 objectives make it clear that C2 products are intended to protect against "accidental" disclosure and modification where all of the users are cooperative. Note that an open network cannot meet the definition of "cooperative users" since there are users who are unknown, inherently limiting the usefulness of such products.

The Division B objective statement [TCSEC85, p.19] introduces the notion of reference monitor and seems to state that even B1 products should implement that concept, and also to imply that the concept need not be realized in C2 (or lower) products. Note that even though the reference monitor concept is mentioned, the principle of simplicity cannot be realized in the context of the requirements of B1 (or perhaps even B2). The class B1 and B2 objective statements [TCSEC85, p.20,26] further expand on the differences between C2 and B1 and limit B1 with new assertions regarding B2. B1 products must include some formalism in their design, e.g., the informal model. Also, the fact that all flaws must be removed (as opposed to the "obvious" ones required at C2) implies a corresponding search for those flaws. Note that, taken to an extreme, one could argue that even identified covert channels need be addressed in the absence of a covert channel analysis requirement. However, one could also conclude that the absence of a covert channel analysis requirement until B2 is an implied exemption for lower class products. The class B2 objective statement implies at B1 and lower, that not all subjects and objects in the ADP system need be addressed, that the TCB interface need not be well-defined, and that the systems need not be particularly resistant to penetration.

From the TCSEC sections "The Trusted Computing Base" [TCSEC85, p.67] and "Assurance" [TCSEC85, p.68] one can conclude that it is imperative that the TCB be identified, including its interface and elements. It does seem that the certainty or accuracy of this determination should improve as the evaluation class increases. It is clear that the policy enforcement mechanism can be distributed in lower assurance products (e.g., C2 and B1), and it is implied that wherever the enforcement occurs it must be analyzed (i.e., evaluated).

From the TCSEC section "The Classes" [TCSEC85, pp.68-69] it seems straight-forward in asserting that the differences between adjacent classes is intended to be substantial. Note that the only essential difference between C2 and B1 products (as realized by TPEP) today is mandatory access control (MAC). Given that prior statements have indicated that C2 and B1 products have similar assurance (i.e., based primarily upon testing) and the fact that the C2 and B1 requirements are very similar, except for the MAC related-ones, perhaps this functional difference is primarily what was intended.

The TCSEC testing guidelines "Testing for Division C" and "Testing for Division B" [TCSEC85, p.86] provide the first definitive indication that there should be an assurance difference between C2 and B1. That is, while the assurance for both classes is primarily derived from testing, the energy devoted to testing the B1 product should be greater than that devoted to C2.

From the TCSEC section "Formal Product Evaluation" [TCSEC85, p.90] there is more emphasis

on testing as the basis of assurance, at least for lower class systems. It is interesting to note that this section indicates that evaluated products may have known flaws. Note that this is contrary to all historical TPEP practices. Furthermore, it is not clear what the consequences of such a practice would be. For example, should such flaws be classified or remain unclassified?

Section 3.2 of the Yellow Book [YBTR85, pp.15-16] implies that a B1 system could protect information in a minimally hostile environment (e.g., confidential data and uncleared users). This is a significant departure from the C2 notion of protection against accidents in a cooperative user environment. These statements also assert that at least a B1 system should be used whenever multi-level data is being processed. In this latter sense, the product is intended primarily to prevent accidents, much like a C2 product.

When taken all together, it can be concluded that the intended, worst-case environments for C2 and B1 systems are subtly but significantly different. That is, a cooperative environment where the product is expected to protect against accidents versus a minimally hostile environment where the product is expected to protect against at least casual attacks.

Environments aside, differences in assurance can be derived from these conclusions. Both may have an ill-defined TCB, but a B1 product must provide an informal model and a reference monitor concept argument. Despite the lack of required TCB definition, the TCB interface and elements must be identified and evaluated for both. Both may have subject and/or objects that are excluded from the protection of the security policy. Both are to be subjected to rather extensive testing, but a B1 apparently should undergo more testing and must involve a search for flaws beyond the obvious. Note that none of these statements alleviate TCSEC class-specific requirements.

3.2. Environments: past vs. present

Statements in the TCSEC and the Yellow Book further substantiate that the authors were primarily concerned about monolithic, closed environment type products. While those may have been predominant at the time, they are now the exception. The products of today are networked and distributed as a rule, otherwise they are not useful in today's marketplace. As a result, the environments of trusted products are no longer friendly, cooperative, and generally benign. Rather, they are exposed to a very broad set of personalities including defense employees, university students, home users, foreign nationals, etc.

From this one could argue quite simply that the need for class C2 type products has significantly diminished. Those products are for use in any existing closed environments, while criteria for a discretionary protection system that is truly resistant to attack does not exist in the TCSEC.

3.3. Environments: intent versus practice

C2 and B1 products typically come with inherent assurance constraints. For example, they are often security retrofits for older existing products, they are developed with a philosophy of "penetrate and patch", and they are not subjected to more rigorous or formal development processes. Except in rare cases, a result is that no matter how many evaluation resources are applied, the obtainable amount of assurance is relatively low (i.e., the point of diminishing returns is reached rather quickly).

It is desired that evaluation teams not attempt to exceed the inherent assurance limitations. Alternatively, it would be worth considering whether it is cost effective to even perform such evaluations in the first place. If such evaluations necessarily have low assurance and that low assurance does not meet today's security needs, perhaps these evaluations should not be performed. There are of course other details to consider, such as whether higher assurance products are available to meet customer needs.

A further consideration is the true number of customers that operate in environments such as those described by the Yellow Book as being appropriate for C2 and B1 class products. A substantial sector of the market (e.g., DIA) has decided that C2 and B1 class products are not suitable for their closed, compartmented-mode environments, and have therefore defined their own evaluation criteria (somewhere between classes B1 and B2, with some extra functions). Many other customers operate C2 and B1 class products in open and potentially hostile environments clearly not intended for these products. The number of customers that operate in truly benign, closed environments is currently unknown to the TPEP, but it is estimated to be small and shrinking. This situation is another reason that continuation of C2 and B1 class evaluation may be questioned.

4. Proposed appropriate evaluation approach

While it is obvious that some amount of analysis need occur to meaningfully evaluate a product, that analysis must be appropriate to achieve the assurance goal. For C2 and B1 evaluations, the analysis should be directed primarily toward the support of testing and secondarily to producing statements that the relevant requirements are met.

It is important to understand that the pertinent analysis must be done by the evaluation team (as opposed to the product vendor) since it is imperative that an "independent" check be made. Note also that the C2 criteria imply a less structured evaluation approach, and it seems that the simple act of defining the level of analysis exceeds the TCSEC expectations.

The following sections represent a simplistic view of the recommended information that must be determined in C2 and B1 evaluations and recommendations regarding appropriate breadth and depth of the relevant analyses.

4.1. Proposed simplistic evaluation requirements

In general, at the C2 level of trust an evaluation team must:

- Identify all protected subjects and objects. (Guidance should be produced which will help evaluation teams and vendors understand what subjects and objects should be addressed in terms of evaluation. This guidance should also identify or specify any relevant techniques for this identification.)
- Identify and characterize the TCB interface. (Guidance should be produced which will help evaluation teams and vendors understand how and to what extent the TCB interface must be identified.)
- Determine how the TCB protects itself.
- Determine how the relevant requirements are met from the TCB characterization, above.

- Test assertions relating to the requirements (i.e., the vendor-prepared tests).
- Extend testing to increase confidence in security mechanisms (note that such extensions may not be necessary if the vendor has a very comprehensive test suite) and to search for “obvious” flaws. Note that analysis cannot substitute for testing. (Guidance must be produced which will help evaluation teams understand how much effort and what resources should be used in identifying “obvious” flaws.)

At the B1 level of trust an evaluation team must perform the functions identified for C2 above and:

- Understand the informal model (prepared by the vendor) and map it to the TCB interface.
- Produce an explanation of how the reference monitor concept is realized, in addition to characterizing the TCB.
- Perform limited penetration type testing to increase confidence in security mechanisms and to search for flaws (e.g., known from other evaluations, including likely vulnerable areas, and hypothesized by the evaluation team as a result of analysis). (Guidance must be produced which will limit and make consistent such efforts.)

4.2. Breadth of analysis

The breadth of analysis for every evaluation class is the entire TCB. Specifically, at C2 and B1 the focus should be on design rather than implementation, and a good characterization of the TCB interface and how the TCB protects itself should suffice. However, at C2 and B1 that interface may be less well defined and will almost certainly contain non-security relevant portions.

After the TCB has been identified, emphasis should be on the security relevant portions and on known problem areas. Known problem areas shall be those known to the entire community as opposed to individual knowledge. Anything that is documented on a community forum, or other community information repository, should be considered known.

The evaluation team need also be able to argue why non-security relevant portions of the TCB are not security relevant, as well as being able to argue that they have considered the entire TCB. A simple assertion to that effect is not sufficient. Basically, something is not particularly security relevant if it cannot be mapped to a TCSEC requirement, other than the system architecture requirement in the sense that it is trusted not to do something outside its design. In general, integrity and function are not security relevant, except as specifically noted in the TCSEC (e.g., label integrity).

4.3. Depth of analysis

The question for depth of analysis is interesting versus uninteresting rather than one of security relevance. Everything in the TCB is technically security relevant, even if it has no effect on the security policy, because everything in the TCB could, by definition, bypass some or all of the security mechanisms. The characterization of interesting will help bound the analysis necessary to make a sufficient argument. For example, if one is basing arguments on design information, the implementation details are generally uninteresting. If the design information is incomplete, however, the implementation details become interesting inasmuch as they are used to reverse engineer the design and thereby make a sufficient argument.

The problem evaluators have today is one of determining whether a detail is interesting (i.e., important) or not for low assurance evaluations. Though there may not be a simple formula to differentiate, some guidance can certainly be given:

- If something is demonstrable with a test, no deeper understanding is necessary. It simply need be tested. It is entirely possible that very little beyond an interface characterization need exist in order to make an adequate testing argument.
- If it does not matter whether or how something works, no deeper understanding is necessary. There are numerous functions that fall into this category. For example, many functions such as file locks are available in ADP systems, but have no bearing on the security policy.
- If something need work correctly in order for the system to work at all, no deeper analysis or even testing is necessary. If it doesn't work, nothing will work.
- Many COTS product interfaces are uninteresting, largely due to the fact that they are not particularly security relevant (e.g., most CPU instructions).

The primary difference between C2 and B1 in terms of depth of analysis should be realized in the area of TCB self protection. This is due largely to the statements identified above that lead one to the conclusion that a B1 product is expected to resist some attacks while a C2 system need not necessarily do so. Also, some extra attention should be applied to the TCB interface in determining what happens when unexpected actions occur. Hence, in a C2 evaluation the team need characterize the TCB interface in terms of the security relevant functions and expected behavior resulting from behaved use (i.e., non-malicious) and characterize the means by which the TCB protects itself. In a B1 evaluation the team need characterize the TCB interface in terms of the security relevant functions and expected behavior resulting from behaved and ill-behaved use (e.g., wrong parameter type) and provide a detailed description of how the TCB protects itself, including an explanation of how the reference monitor concept is realized.

In those cases where there is insufficient design documentation to complete an argument (e.g., characterize the TCB interface), the evaluation team must either derive the argument from a lower abstraction or fail. Both are undesirable; therefore, the vendor should be required to produce the appropriate evidence. Note that the TPEP guidelines "Form and Content of Vendor Design Documentation" and "Form and Content of Vendor Test Documentation" are intended to describe such requirements for the current TPEP evaluation process.

5. Recommendations

Though the TPEP should consider whether C2 and B1 evaluations should be continued, the following recommendations are offered under the assumption that they will continue.

- The evaluation approach, including breadth and depth guidance, of section 4 above should be considered by the TPEP and expanded with detail, guidelines, and examples to provide a workable model for such evaluations.
- The TPEP should provide direct technical oversight to ensure that evaluators are performing appropriate analyses in terms of breadth and depth (e.g., effective Senior Evaluators and Technical Leaders, direct technical oversight and guidance, and extended TRB involvement). The people doing the oversight must be highly experienced and technically astute, unafraid to

speak up, and, as a group, in constant communication with each other. They must also be willing to impose, or work within the confines of, community and management decisions.

- The TPEP should ensure that appropriate time is spent at milestones to correct identified errors so that they will not recur and become pervasive, as has happened in the past with level of analysis. This necessarily means that some emphasis need be placed on fixing problems as opposed to the current driving emphasis on completing individual evaluations as quickly as possible. It is very likely that the cost to one project to fix identified problems will likely result in a savings for subsequent projects.
- The TPEP should produce worked examples of appropriate and inappropriate depth of analysis.
- The TPEP should seek to define a list of problem areas expected to be covered in breadth and implement a procedure to keep it appropriate and current.
- The TPEP should make sure that all of their guidance documents (e.g., subject/object and TCB identification) provide appropriate and useful information that would help evaluators and vendors understand the relevant differences between the various evaluation classes (e.g., B1 vs. B2).
- Most importantly, the TPEP should contribute to and consider the Common Criteria to promote a smooth transition (perhaps by redirecting and tailoring current processes for alignment) and to ensure that the Common Criteria future will not include problems similar to those identified here.

6. References

[TCSEC85] Department of Defense Trusted Computer System Evaluation Criteria, DoD, DOD 5200.28-STD, December 1995.

[YBTR85] Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements, DoDCSC, CSC-STD-004-85, June 1985.

MEASURING CORRECTNESS AND EFFECTIVENESS : A NEW APPROACH USING PROCESS EVALUATION

Klaus Keus
Klaus-Werner Schröder*

Bundesamt für Sicherheit
in der Informationstechnik
Postfach 20 03 63
D - 53133 Bonn
Germany

Abstract

Today, an Information Technology (IT) system or product can be evaluated against a number of well known security evaluation criteria. Several governments have released such security criteria, and there are some substantial differences in the application of that criteria. On the one hand, there is a tendency to unify security evaluation criteria, on the other hand new techniques are looked for to enhance security evaluations with respect to time and cost effort. There is a lot of discussion on how the development methods could be taken into account to reduce the effort of security evaluations. Many open problems remain to be solved in this area. Furthermore, conditions need to be defined on how to identify development methods, and tools as well, possessing the potential power to generate IT systems or products of high quality with respect to security needs.

Typically, security evaluations are carried out in a product (or system) oriented manner. Another type of evaluation which is more process oriented is well known as a means of quality enhancement and quality control. As a paradigm, a well known and evaluated process should lead to a high quality process output. Thus, in this paper the approach is to look for connections between product oriented evaluations and process oriented ones. The main goal of such an approach should be to reduce evaluation efforts by incorporating results of a process evaluation into the evaluation scheme focussed on the IT system or product under evaluation. For example, what results could be incorporated and how this could be done differs from hardware to software. Furthermore, it seems to be not too difficult to answer questions with respect to correctness aspects of a security evaluation. The aspect of effectiveness is by far more complicated to handle. But, more research is needed on the subject.

* e-Mail: keus@bsi.de, kws@bsi.de

1. Introduction

For more than ten years evaluations of Information Technology systems or products with respect to their security properties have been carried out. Such evaluations were based on well known security evaluation criteria [4]. The criteria have their origin in governmental security needs. Nowadays, there is also a need to evaluate commercial off the shelf products with respect to security features they provide. The goal of a security evaluation is to approve the trustworthiness of a system's or product's proclaimed security properties. Two aspects of trustworthiness must be considered, namely the aspects of correctness on one hand and of effectiveness on the other. Therefore it is necessary to measure both aspects in terms of mature and well accepted criteria. Different criteria could be used depending on the focus of an evaluation. There are two types of evaluation criteria. One type is focussed on the special system or product under evaluation (product oriented approach). The other type concentrates on the investigation of the development and production process (process oriented approach). As a consequence, this paper poses the question what the relation between these two different approaches should be.

In section 2 some basics of product evaluations are sketched. The goals of correctness evaluation as well as of effectiveness evaluation are identified. A short comparison of product oriented and process oriented evaluations is given in section 3. Process evaluation attracts more and more attention in conventional areas of industry. Nevertheless, the quality ensuring measures which are typical for product checking, are applied as well.

How could evaluations of a product or of a process be applied in the young field of information technology? This is the question of section 4. It is revealed that hardware production and software production differ totally from each other. To produce hardware all the well known methods and production steps are applied that could easily undergo a process evaluation. With respect to software production there are some problems and open questions. As far as software is concerned the interactions between product oriented topics and process oriented topics of an evaluation incorporating both aspects are rather unknown. Some remarks are given on how to treat correctness. A perspective on effectiveness is given in section 6.

2. Basics of Product Evaluation

Security evaluations of IT systems or products are carried out in Europe and in North America. The criteria used as a bases of such evaluations are the harmonized European criteria [1], the Canadian criteria [2], and the US-american criteria [3].

An IT system or product to be evaluated should have well defined features to protect against threats or to assert a certain security policy. The main goal of an evaluation is to get an independent confirmation that the promised security properties hold. Prior to this confirmation the IT system or product is to be checked on the basis of the criteria. Evaluation is a means to increase confidence in the security achievements of Information Technology. It is well known that trustworthiness splits off into the two aspects of correctness and effectiveness. Both aspects have to be investigated.

Roughly speaking the aspect of **correctness** evaluates whether an IT system or product corresponds to the design. As a consequence all the possible incorrect transformations of a top

level design into an implemented product can be recognized during correctness analysis. Thus, proof of correctness may be viewed as a trial to minimize differences between design and implementation (oscillating behaviour of a process). But, because of differing conditions in the framework even comparable approaches may lead to incomparable differences. Another goal of correctness analysis is to evaluate the development methodology with respect to error avoidance or robustness. The statement resulting from the analysis should be that the implementation of the IT system or product under consideration *is correct* with respect to the design.

The aspect of **effectiveness** should evaluate whether the means of a system or product, respectively, are appropriate to prevent from threats, and to assert the security policy. Effectiveness analysis is not part of all the security evaluation criteria mentioned above. Criteria that know the notion of effectiveness clearly distinguish it from correctness. Moreover, although effectiveness is based on correctness to a certain extent, effectiveness analysis is to be carried out independent from, and in addition to, correctness analysis. A major goal of effectiveness analysis is to detect weaknesses as well as shortcomings in the intended action of the IT system or product which may be part of the design. Of course, correctness analysis would not detect them. As a consequence, effectiveness analysis can reveal possible conditions under which the system may not serve the security needs. As a result it should be stated that the design *provides suitable means* to assert the security policy.

In combination both aspects lead to a statement that the protective action will be achieved by suitable means which are correctly implemented (cf. fig. 1).

The definition of security goals is a precondition to an evaluation. In case of a product these goals are called *product rational (PR)*, in case of a systems the notion of a *security policy (SP)* is used. All the investigations carried out to evaluate both aspects of effectiveness and correctness are based on the PR/SP.

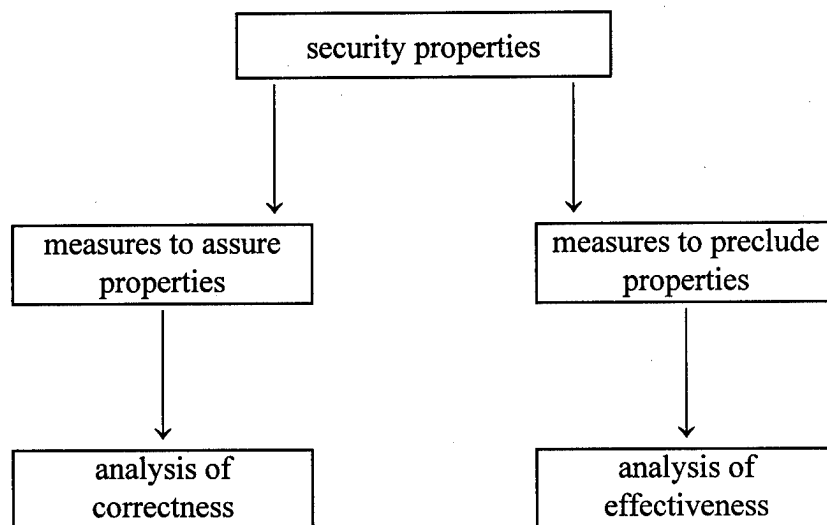


Figure 1 : Product evaluation as a measure to assure security properties on the one hand, and to preclude other properties which undermine security on the other hand.

It is obvious that, in addition to attacks on an installed system, attacks to the development process itself, are possible too. Therefore, it should be natural to include the development as well as the production processes into an evaluation. One may think of systems strongly depending on a secret of the developer to achieve the security objectives, but also, unrecognized alterations of the system or product may lead to a lack of security. Therefore, security (as a combination of confidentiality, integrity, and availability) of the developmental data should be made a point to be evaluated, too.

Checking the developers security as a task of the evaluator suggests that there is a connection between product evaluation and process evaluation, although these two types of evaluations basically differ from each other. The differences arise especially in regard to statements concerning the product itself, taking into account that the final goal of process evaluation is to give a statement on a product.

3. Basics of Process Evaluation

In contrast to product evaluation where the primary goal is to get confidence in the specific security properties as derived from the implemented security functions of a single product process evaluation addresses the more general demand for high quality products as stated by vendors as well as by customers. In process evaluation the evaluator draws a conclusion from a single result, e. g. based on a random sample, to the properties of products produced by the process under consideration. There is a certain probability that the conclusion holds. It depends on the procedures used which may be based on time, number, or other relevant measure.

Process evaluation has a long history. Some of the milestones may be of interest in order to get an impression on the features :

- development of instructions on how to make a product (craft skills, quality maintaining)
- use of (mechanical) machines (reproduce within known tolerances, quality generating)
- introduction of checks during as well as at the end of a production process (confirmation to be within the limits, quality maintaining)
- establishment of vocational / professional training (control increasing complexity, quality generating)
- random samples as a means of quality control (reduction of cost and effort, quality maintaining)
- increase of motivation and raising of people's awareness (total quality management, quality generating)

Of course these items don't cover the whole range of quality measures. The primary goal of the measures mentioned above seems to be a production process with inherent high quality. It is hoped then that all products also show high quality properties. Whatever the functional properties may be there is a well defined quality level.

There is much discussion on process evaluation. It seems to be a further means within the range of quality measures. As stated such measures can be subdivided into two classes generating quality on the one hand and maintaining it on the other. Of course they may be used in

combination. Process evaluation seems to be a means to identify quality generating steps of a process (cf. fig. 2). It should be remarked that a certificate of process evaluation is valid only within a restricted time range. Therefore, re-evaluation of the process is necessary.

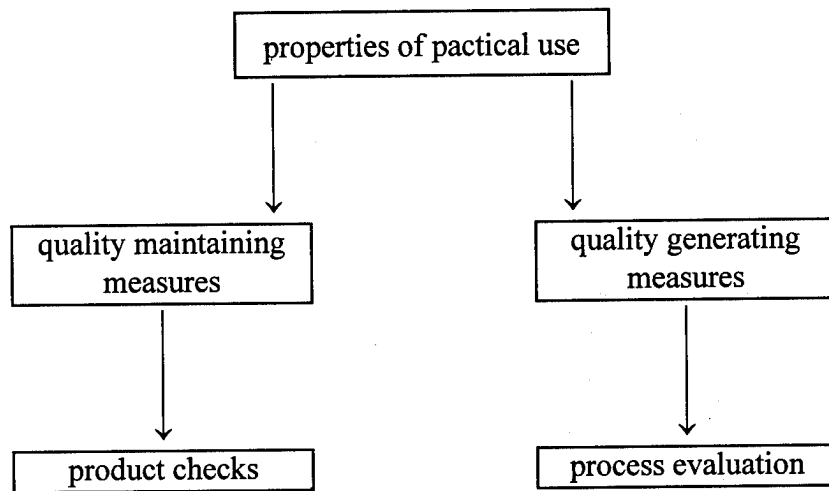


Figure 2 : Process evaluation as a quality generating measure to assure practical properties of products.

4. Information Technology : Hard- and Software

The simple comparison of figures 1 and 2 illustrates the fact that the evaluation of products and the evaluation of processes use different methods. Although both procedures have the common goal: "Improve the creation of assurance", the primary results are very difficult to compare. The situation becomes more complicated with respect to the combination of hardware modules and its specific tailored software modules (cf. fig. 3). Putting product evaluation opposite process evaluation in information technology will reveal differences in the development and production processes of hardware and software.

The development process and the production process of IT-hardware (e.g. integrated circuits) are very similar to the production of other non-IT products. Differences exist concerning complexity and accuracy. The design phase will be followed by the production phase. Based on the use of CAD-tools the layout will be created, which will be tested with respect to well defined procedures and rules before prototypes are produced and tested. All design based errors will be recognized by such quality ensuring steps. However there exist unavoidable and production based errors, which often have physical background based reasons. During the production process specific quality ensuring procedures are used to satisfy given tolerance allowances. One goal of all these kinds of quality activities is the proof of the correct implementation of the layout. As far as that is concerned, the evaluation of the process will contribute in a positive way to the aspect of correctness because of quality building activities and quality ensuring aspects.

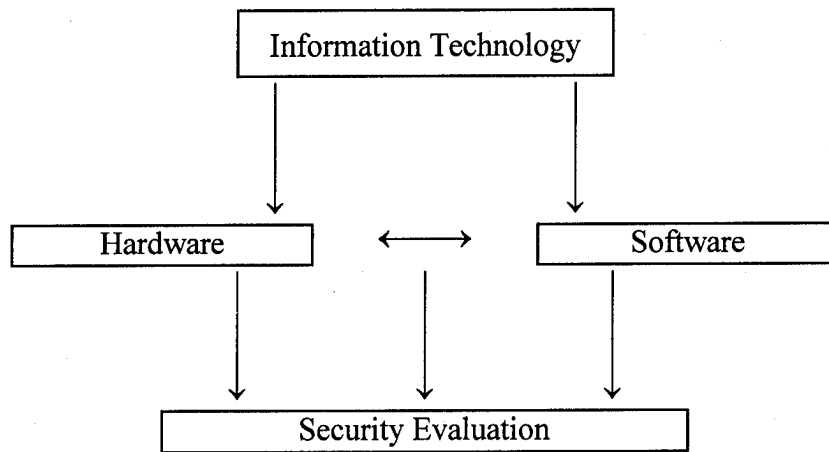


Figure 3 : Security evaluation of Information Technology links three aspects together : hardware, software, and their interface.

It is important and necessary to have a clear and visible relation between the application properties and the security properties of the IT-hardware products to use the evaluation results of assurance correctness as input and part of the security evaluation for its application properties.

Further analysis is required to express and explain which steps of the process evaluation may support and increase the evaluation based on products.

In respect to the fact neither having coordinated criteria for these aspects nor using coordinated security evaluation criteria, initial experience has to be gained at low assurance levels to get the first restricted statement and to have a chance for a first survey. It should be possible to expand the experience based on single cases to more global statements.

The software development process including the design phase and its production process are much more complicated than the more general production of IT-hardware. This is not very surprising because of the young tradition in software development and the attendant restricted experience. The main difference to IT-hardware is the fact that all relevant errors are based on design. Software production is limited to the reproduction of digital copies using a prototype, which is expressed by the notion of *master copy*, excluding the fact of differences between the prototype and the final product. Production oriented errors (e.g. errors or mistakes between copying, material based errors in the results) will be recognized during the process and will be removed using quality building steps. Hence the main focus has to be taken in the design phase.

Unfortunately the notion of software design belongs to the class of poorly defined expressions. It is not quite clear how to distinguish between design and implementation, especially in separating the implementation phase from the production phase. The use of code generators and the reuse of modules will reduce the clearness of their limits. As during the production of IT-hardware components all the different phases of production have to be performed for each product, there is one complete process for the production of software. The software product is built completely different from the hardware product and its production differs completely from the production of its prototype.

5. Result

Regardless of all the differences concerning the means and methods used, both approaches have one common goal: the creation of trustworthiness to the product, including hardware and software. As of at least today, the results are not comparable at a first glance, so the most important aim is the combination of both procedures in one common concept towards a more efficient solution. Arising questions are :

- Is there a way to built further statements on the foundation of a qualified and IT-security specific evaluation of the process or at least to use the statements of the process evaluation (quality maintaining and quality generating) as input for the process of product evaluation?
- Which kind of requirements concerning the security aspects may be transmitted to the process?
- The required specifications for IT-security, do they need to be tailored and translated to the specific requirements or to the specific notions of processes?

The first interesting and more promising approaches are available, see refs. [5] - [10]. But there is still a need for more detailed and practical support and operation including aspects as to "who", "when", "where", "how", "what" and "why".

6. Forward / Perspective

The link between the evaluation of assurance effectiveness based on the process evaluation and its advantages is much more complicated than the correctness part. A basic requirement is the improvement of the evaluation of assurance effectiveness concerning its structure, e.g. based on a metric using hierachical and rated evaluation steps, binded with tailored evaluation methods and procedures or rated evaluation profundity. First approaches may be comparable to the solution for the strength of mechanisms in the ITSEC [1] or to the evaluation in the assurance correctness part. Afterwords the analysing phase of the process evaluation will be started with respect to a suitable combination of assurance effectiveness.

The PR/SP as the basic scale of the evaluation of the assurance effectiveness has to be integrated into the process as the main starting and focus point, i.e. the PR/SP has to be respected during the development and the production in a way that it has direct technical significant influence to the effectiveness of the product as part of the process. This approach requires the definition of a concrete and strictly defined PR/SP, e.g. concrete operating environment or the definition of concrete threat scenario etc. Hence it has to be followed by a strictly PR/SP-depending process, e.g. defined by the use of specified tools, models, and SE-rules. Additional definitions for all the single and all the detail phases and procedures, clear defined structures including the significant definition of the complete process and its single phases, their interfaces, and their traceability have to be given. Rules concerning the methods and procedures for validation and verification as well as for the single processes and their interim results need to be defined. Questions will arise as

- Which kind of overlapping or differences will occur based on an object oriented approach compared to the more general waterfall model?
- If the evaluation of a process will include the parts of the assurance correctness along with parts of the assurance effectiveness, which of the specific requirements will exist in respect to the combination of both evaluation concepts as input for a single common solution?

Basic preventive conditions have to be defined as suitable SEU, equipment, organisation, models, CM, PM, questions concerning the competence of developers etc. This requires a more global structure based on the analysis of all the relations created by using all the different aspects (dimensions) as parts of a more common metric in the sense of equivalence classes.

References

- [1] Information Technology Security Evaluation Criteria (ITSEC), Version 1.2, June 1991
- [2] The Canadian Trusted Computer Product Evaluation Criteria, (CTCPEC), Version 3.0e, January 1993
- [3] Common Criteria for Information Technology Security Evaluation, Draft Version 0.9, October 94
- [4] Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, Department of Defense (1985)
- [5] Paulk M.C., Curtis B., Chrissis M.B., Webber C.V., Capability Maturity Model for Software Version 1.1., February 1993 Software Engineering Institute
- [6] Trusted Capability Maturity Model for Software, NSA / Software Engineering Institute
- [7] Security Engineering Capability Maturity Model for Software, NSA / Arca Systems Inc. / Computer Sciences Corporation
- [8] ISO 9000: Quality management and quality assurance standards
ISO 9000 / 3: Quality management and quality assurance standards; Part 3: Guidelines for the application, supply and maintenance of software
ISO 9001: Quality systems - Model for quality assurance in design / development, production, installation and servicing
- [9] Haase, Messnarz Koch, Kugler & Decrins: Bootstrap: Fine-Tuning Process Assessment, in IEEE Software, July 93
- [10] Keus K., Kurth W., Loevenich D.: IT-Security: - a Quality Aspect ! Quality Assurance in the ITSEC-Evaluation Environment in Germany, in proceedings to the National Computer Security Conference, 1993

Reengineering the Certification and Accreditation Process: Security is Free.

Sean G. Mahon
Boeing Information Services
7990 Boeing Court
Vienna VA 22182-3999
Mahon@Dockmaster.NCSC.MIL

"If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization."

-anonymous

Abstract

The purpose of this paper is twofold: first, to show that Certification and Accreditation can be a value-added process that can help improve system security. Second, to show that security, when it is considered part of the system design, does not increase the cost of the system.

Key words:

Accreditation, Certification, process, quality

Most Departments (e.g. Defense, Treasury, Energy) and agencies of the US Government require that their Automated Information Systems (AIS) that process sensitive or classified information be certified and accredited. Yet, most organizations try to avoid certification and accreditation. Studies have shown that half of DoD Systems are not accredited or have had their accreditation lapse without being reaccredited. (AISSD,1991) (Jaworski, 1994) Security in computer systems is viewed in the same vein, a necessary evil, that is required by policy and generally gets in the way of doing business.

This is exactly how quality assurance was viewed by General Motors and Ford in the 1970s. W. Edward Deming, Phillip Crosby and most recently Hammer and Champy have shown that those views are not valid in the competitive environment of the 1990s.

There are two major assertions to this paper. First, that Certification and Accreditation can be a value-added process that can help improve a system instead of being a meaningless paper drill. Second, that security

when it is considered part of the system design does not increase the cost of the system.

Returning to the automobile analogy, there was a massive shift in the public's view of what constituted quality in automobiles in the late 1970s and Detroit had to catch up with the Japanese auto makers. Today, the countless viruses and internet attacks are reshaping the public's view of what constitutes security in computer systems and networks. Philip Crosby, in his book Quality is Free maintained that quality is free because when you consider the cost of waste and rework due to the lack of quality, effective quality control saved money. The lack of quality was more costly to a company than having effective quality control. Quality pays for itself. The book's title comes from a quote by Harold Geneen, formerly the head of ITT: "Quality is free, it is not a gift, but it's free."

Similarly, the cost of responding to the lack of security in today's information systems is increasing daily, and security has become necessary to the survival of systems and networks. Today, the vast majority of systems are not very secure and the costs of cleaning up viral outbreaks and cracker attacks are mounting. Retrofitting security into systems, (i.e. investing in firewalls) would not be necessary if security were more than an afterthought in system design. A recent article on a hacker attack in TIME magazine illustrates the changing environment.

"Across the country computer network security experts were calling the entire Mitnick affair a watershed moment-not only for what it proves about the hacker but for what it says about the systems he hacked. At a time when American businesses are frantic to set up shop on the computer networks, those networks-and the telecommunications systems that carry their traffic are turning out to be terminally insecure."

The public attitude shift toward car quality in 1975, caught General Motors, Ford and Chrysler flat footed. The operating system vendors such as Microsoft and Novell, system integration firms, as well as the Government can learn from their example and begin to practice good security engineering in acquiring and maintaining their systems before the lessons become more painful.

While the term "Reengineering" has been coined by Hammer and Champy, the idea is not new. "Starting from zero" as a method of design was preached by Walter Gropius of the Bauhaus school of Architecture in the 1920s (Wolfe 1982). Systems theorists have said for years that automating bad manual systems result in bad automated systems. However, Hammer

and Champy, by defining these activities with the phrase "reengineering," have forced organizations to look at their current processes and see if they are the shortest most efficient path to their corporate goals.

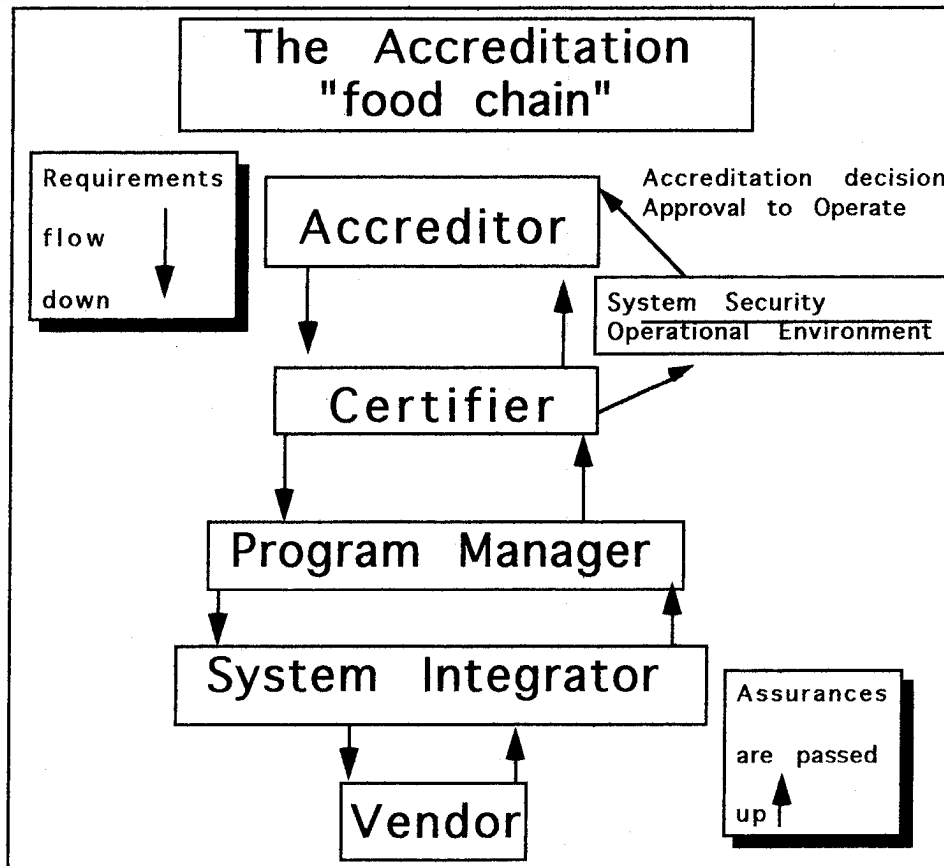
"Reengineering is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical contemporary measure of performance, such as cost, quality, service and speed." - (Hammer and Champy, 1994)

In order to reengineer anything you must have a goal that you are trying to achieve. The goal of the C&A process should be to build and maintain certified and accredited systems, with the requisite level of security, that users can use to do their work. The last part of the goal is the most critical, because when all is said and done, someone must use the system that you certify and accredit to do their job every day. This should be the central vision of the C&A process, not the DAA's risk exposure. What good is a system with zero risk, if it makes life for the user so difficult that it renders the system useless?

The C&A process defined:

"Steps - the processes are formed in a natural order"
- Hammer and Champy.

For the purposes of this paper, the C&A process that will be modeled is based on a generic (Army Regulation 380-19) or type accreditation (Air Force Regulation 205-16). Operational and site based accreditation are focused on the delivered system, and are geared to uncover vulnerabilities after installation, which is the most expensive point in the life cycle to address them. The system security certification, as a process, should parallel the system development cycle and be fully integrated into the system test process. In this manner, the data needed to make an assessment of the system's security, is developed along with the system. The steps leading from system design to certification and then to the accreditation decision should be a natural progression.



The Accreditation Food chain: Each entity in the chain receives products from one level in the chain and passes on products to the next level. It is convenient to start with the vendor, since it is the vendor that supplies the basic components on which integrated systems are built, namely hardware platforms and operating systems and other commercial off the shelf packages. Requirements flow down the chain, assurance is passed up the chain. If the C&A process is started in conjunction with the system design process, and C&A requirements are flowed down the lowest level, then the system should be almost be self certifying.

The vendor's product is used by a system integrator as a component in a integrated system. This system (or a proposal to build this system) is offered to a Government Program manager who will oversee the integrator's efforts.

The Program Manager will have to find a certifier for the system and an accreditor to assume risk and allow the system to operate, if these entities have not been already designated.

Everyone in the food chain has a part of the C&A puzzle. In order for the process to work, everyone must come to the table with their piece of the puzzle.

The puzzle pieces:

The Vendor : Provides insight into the inner working of the COTS product.

The System integrator: Understands the System Architecture, how the components fit together, how the system security policy is enforced through the various system components.

Certifier: Evaluates the system security in regard to the operational environment, quantifies residual risk for the accreditor

Program Management Office: Identifies/validates users needs, ensures system meets functional requirements. Represents the voice of the user.

Accreditor: Makes decision to allow the system to operate.

How it should work:

The Program Management Officer (PMO), as part of the Request For Proposal, asks bidders to state in their proposals how they intend to support the certification and accreditation process.

Integrators bidding on the proposal, propose a set of assurance evidences that they will supply to support system certification.

When the contract is awarded, the proposed assurance package is fine tuned through technical exchange between integrator, PMO, certifier and accreditor (or accreditor's representative). If all of the parties approve the approach and meet regularly to discuss issues, the accreditor is no longer asked to make a "leap of faith" at the time of the accreditation decision.

Once the accreditation has been achieved, the same process should be followed with each major change to the system. If certification requirements are considered when changes are made, then the recertification and accreditation will be merely review of the evidence produced as a part of all the changes made to the system since the last accreditation decision. This process would be similar to the RAMP

methodology that is used for evaluated products. A security analysis would be conducted for each engineering change order, and evidence of the certification maintenance would be retained for the certifier's review.

Adding value to the process

What vendors can do:

Vendors should be proactive, and ask questions on product use and C&A support the integrator will require. If your product is in NCSC evaluation, have your vendor security analyst (VSA) talk with the Integrator's security engineering staff to ensure that the system makes best use of the product's security mechanisms and the product is being correctly presented to the certifier.

Help your customers (Integrators, Government PMOs) understand the security your product provides, especially if it is an evaluated product. Many integrators and PMOs do not understand the value of having an evaluated product and what the evaluation means and what it does not mean. To most, it is simply a box to be checked off.

Try not to let your marketing staff stifle technical interchange between the vendor and the integrator. Feedback from your customer can be used to make your product better.

Adding value to the process

What System Integrators can do:

As the builders of the system the integrator can make the system practically self-certifying by managing and packaging assurance evidence for the certifier to review. Assurance evidence management (Arca 1994) practiced by the integrator will reduce the level of effort needed by the certifier.

Adding value to the process

What Government Program Managers can do:

Verify the users requirements. Find out what trade off the users are willing to accept. Users want and need some level of security in their information systems, no one wants their documents altered or privacy violated. The program manager must be the conduit of information between the system integrator and the user, particularly in regards to design tradeoffs. When all the system requirements are totaled up, there may be no solution space left. It is then the hard lot of the PMO to work

with the user and the system developer in the realm of the possible, to deliver the best fit solution.

Adding value to the process

What NSA and DISA can do:

Expand the education process so that system integrators and government PMOs can understand the Trusted Product Evaluation Process (TPEP) and what it means to have a product evaluated. The dialogue and vocabulary is now well established between the TPEP product vendors and the NCSC (C71) evaluators. Disseminate that knowledge.

Some suggestions on accomplishing this:

Have a Trail Boss course for acquiring trusted systems , to educate government acquisition personnel in the issues involved in building trusted systems. Open the VSA course to integrators and government PMO personnel.

A Case study in reengineering C&A:

This case involves a large MLS system in DoD. After contract award, disagreement arose over what assurances were required to certify the system and who should provide them. The program manager and the certifier found it difficult to express exactly the form and nature of the assurance. The disagreement centered primarily on the operating systems that were being used, which were in the very preliminary stages of NCSC evaluation. Many meetings were held and correspondence flowed between the Government program manager and integrator, integrator and vendor. Eventually an approach to gaining the required assurances was hammered out, however only after many resources were expended in non value added activities such as writing and replying to contract correspondence.

Later in the system development process a multi-level e-mail feature was to be added to the system. This time the integrator ensured that there was concurrence from the certifier and the program manager as to what assurance evidence was required for the multilevel e-mail product. These requirements were flowed down to the vendor, where applicable. The integrator and the vendor provided the certifier with a security design briefing and security design documentation. A member of the certification team witnessed the integration testing and leveraged that into better security test coverage for the government acceptance test. The end result: a better certification of the system, no expenditure of resources in non value added activities, and there was no increase in price of product or

the level of effort for integrating the product in the system.

Rules for applying reengineering in the C&A process.

Rule 1: Security is cheaper, better, less intrusive and more effective when you design and build it into a system instead of retrofitting.

Rule 2: Treat security as necessary functionality. Just as you must give a user today a GUI interface and a WYSIWYG word processor, you must provide some level of security, as a minimum I&A, file access control, and data integrity.

Rule 3: For C&A "generic" or "type" accreditation is more cost effective than operational or site based accreditation.

Operational and Site based accreditations are geared to identifying vulnerabilities in systems after they are fielded, when it is most costly to correct them. It makes more sense to put the onus of ensuring adequate system security on those organizations responsible for the systems design and acquisition, rather than the operational user. Certification should start when the system design starts.

Rule 4: In system changes and maintenance, ensure that security and C&A requirements are considered before implementing change.

Rule 5: Know thy accreditor. Bring him in early in the system design process and keep him informed.

Rule 6: Assurance is where you find it.

If the system is built and maintained in accordance with sound system and software engineering practices, there are security assurances already in your processes. Process assurance is free. The certifier should have insight into how the integrator is building the system.

Rule 7: Be flexible in your design, requirements and environments change and specifications become outdated and irrelevant. Don't allow security design tradeoffs, made earlier on, to paint you into a corner or tie you to dead end technology.

Rule 8: Hope is not a method.

Your accreditation effort should not hinge on the hope that the accreditor will accept unusually large risks on blind faith or a pile of documentation you present as an end product. Second, do not allow yourself to think that the new technology that is trumpeted in a glossy brochure is going to put a

end to your security problems. There are no silver bullets. Careful analysis of the security requirements and the design constraints will more often lead to a solution, than adding a security black box to the system.

Rule 9: Your security engineering staff cannot be everywhere at once. Every member of your organization should be sensitive to security concerns. This can only happen through training and good organizational communications.

Summary

Meeting the goal of building and maintaining a certified, accredited and usable system is not easy. Many systems today are caught in the current Bermuda Triangle of government specifications and standards, e.g. GOSIP, GUI, MLS, EPL, COTS, Ada, PCMCIA, X.everything, DMS, DSS, WYSIWYG. There are no solutions that will meet all these standards and specifications. Tradeoffs have to be made. The process and principles outlined in this paper should assist all parties involved to find the best fit solution that provides the balance between functionality and security. Second it should help reduce the cost and resources involved in the C&A effort. There is not enough empirical data to prove the assertions made here, but the initial cases have shown much promise.

References

- Arca Systems and the National Security Agency (1994) Security Engineering CMM Straw man.
- Automated Information System Security Detachment (AISSD) US Army "1991 Computer Security - Lessons Learned"
- Bauer R. et al, "A Framework for Developing Accreditable MLS AIS" (1991) Proceedings of the 14th National Computer Security Conference.
- Hammer M. and Champy J. (1993) Reengineering the Corporation. New York: Harper Collins
- Jaworski L., "The Network Memorandum of Agreement Process: Lessons Learned"(1994) Proceedings of the 17th National Computer Security Conference.
- National Computer Security Center. (1985) Department of Defense Trusted Computer System Evaluation Criteria Washington D.C.: US Government Printing Office.
- Sachs J. et al, "What Color is your Assurance"(1994) Proceedings of the 17th National Computer Security Conference.

Wolfe T., (1981) From Bauhaus to our House, New York: Pocket Books