What Might We Mean By "Secure Code" and How Might We Teach What We Mean?

Cynthia E. Irvine
Naval Postgraduate School
Monterey, California

Abstract

To add precision to the term "secure code," several concepts from the area of high assurance system development are introduced. These allow the cost, in terms of organizational damage, to information asset owners of the lack of various forms of protection to be explored. The notion of system assurance levels is introduced as a solution to the combinatorial explosion of damage classes. Our approach to ensuring that students know how to construct evaluatable components, and as a by-product, "secure code," is discussed.

1 Introduction

The word "secure" has a number of definitions: some for an adjective and others for a verb. For the adjective, a notion of protection against danger is associated with the word. To understand danger, it is necessary to determine the value of the entity in harm's way and how it might be vulnerable. It is then possible to determine how to reduce the danger. A fence of a certain height with a post and rail spacing of some maximum dimension will protect a flower garden against flower-loving deer. The same "secure fence" will not protect the same garden from rabbits, gophers or beetles.

If the danger is inchoate, then what does is the term "secure code" intended to mean? Is it code that protects against something? If so, what? Consider the following protection objectives:

- Prevent unauthorized use of personal information.
- Ensure that elections are valid.
- Prevent unauthorized use of entertainment media.

What is the relationship between these protection objectives and secure code? If these abstract objectives can be translated into technical objectives, how does secure code contribute to the achievement of technical goals?

The purpose of this paper is to discuss a possible interpretation of the term "secure code" and to present several approaches to teaching university-level students concepts that will result in systems that achieve protection objectives. These approaches have all been used at various times in the context of several synergistic courses, and constitute a cornerstone for a large-scale security project. Challenges for the instructor will be discussed, as well as solutions we have adopted.

To begin this exploration, some concepts and terminology are presented.

1.1 Trusted and Trustworthy

In the past computer systems evaluated against security criteria were called "trusted". [1] Although a current term of art within the security community, reality is quite different. Any system that the end user places confidence in, whether that confidence is warranted or not, is trusted. Some might trust a deer-proof fence to keep the rabbits out, but that trust is unwarranted. Thus a precise definition is needed:

> **trusted:** the degree to which the user or a component depends on the trustworthiness of another component. For example, component A trusts component B, or component B is trusted by component A. Trust and trustworthiness are assumed to be measured on the same scale.[7]

A term is needed to describe components that merit trust. For this *trustworthy* is used. The dictionary definition of *trustworthy* is: worthy of confidence [2] In the context of computer systems, the meaning can be refined:

> **trustworthy:** the degree to which the security behavior of the component is demonstrably compliant with its stated functionality (e.g., *trustworthy component*).[7]

In Section 1.3 the question of the extent to which as system is trustworthy will be addressed. However, it is first necessary to appreciate why systems and software need to be trustworthy and how this need might be represented in a format that could be translated to a technical implementation.

1.2 Information Assets, Threats, and Policy

Computers and networks are used to manage information; an individual or an organization uses one or more computers to create, modify and manage information assets. Networking allows information assets to be moved among the computers. Practically everything on a computer can be interpreted to be an asset including the operating system, applications, and user data. In the case of user data, assets may range from a bit of humor sent via email, which is likely of little value, to highly valuable trade secrets and financial information. Recognition that information assets have value and that they must be protected from various risks results in the formulation of a information security policy.

Information assets must be managed so that some degree of availability, integrity, and confidentiality are provided for them. Here we do not attempt to refine the definitions of these protection characteristics, but note that integrity is particularly complex [20]. Little protection is required in the case of the "joke of the day," whereas corporate accounts clearly require handling that will ensure some level of protection.

It makes no sense to attempt to formulate a separate security policy for each information item. Instead, information is organized into broad classes and policies are applied to those broad classes.[1] All information can be characterized by some consequence that will result in the event that its integrity, availability and confidentiality are not preserved.

Consider two hierarchical levels of damage that could be inflicted upon a system for each of the integrity, availability, and confidentiality of information assets, as shown in Table 1.

[1] Note that an individual data item may have different value to different people or organizations. These differences are generally resolved through treaties, public policies, laws, and other non-technical vehicles.

Table 1. Levels of Damage to System

Level of Damage	*Integrity*	*Availability*	*Confidentiality*
Catastrophic	CD_I	CD_A	CD_C
Low	LD_I	LD_A	LD_C

Since these are characterized as equivalence classes, they can be organized into lattices [13], and since the Cartesian product of two lattices is another lattice [9], various combinations of the characteristics result. This is illustrated in Figure 1.

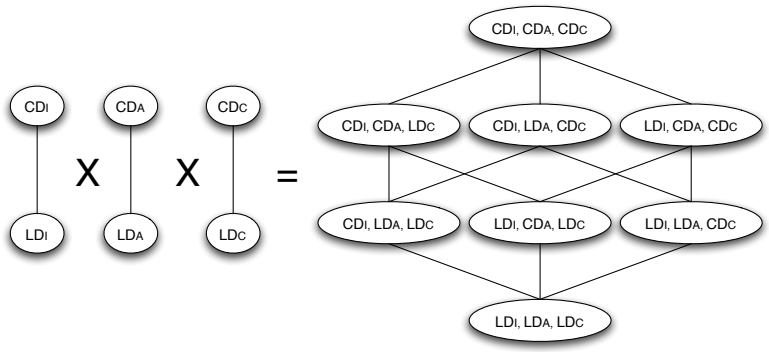

Figure 1. Damage Characteristic Equivalence Classes

The symmetry of the lattice on the right hand side of Figure 1 is illusory. Many of the classes are non-comparable and a less visually pleasing, highly distorted graph would be equally valid. The non comparability of the damage classes is important since it ultimately leads to a combinatorial explosion of the levels, as will be shown later.

At the top of the resulting lattice is information for which the consequences of any breech of its integrity, availability and confidentiality would be catastrophic, and at the bottom is information for which such events would be of little or no consequence. Should only one more equivalence class be added, for example Moderate Damage to information Integrity (MD_I), four more nodes must be added and the lattice becomes considerably more complicated, as shown in Figure 2.

The organization of information allows system and software developers to understand the degree to which a system component will be relied upon to perform both correctly and only as specified. The characterization depicted here reflects not only the explosion of labels envisioned by Lipner for commercial environments[30], but adds availability as a dimension to the information's characteristics, further increasing the combinatoric expansion.

It is worth noting that availability has traditionally been the neglected stepchild of the triad of information security characteristics. This is due in part to the fact that the more subjective nature of availability renders it less amenable to traditional formal modeling techniques, although notable work in this area exists, e.g, [41]. An example of the availaibility problem can be seen in scheduling: those requiring a real time system will have radically different availability requirements from those who require a traditional round-robin scheduler.

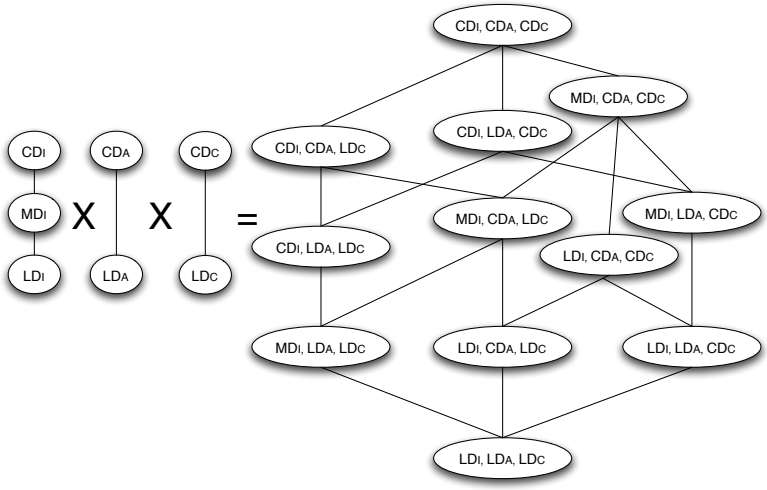

Figure 2. Extended Damage Characteristic Equivalence Classes

1.3 Protection: How Much is Enough?

A fundamental question to be answered is how much trustworthiness is required in a particular system or component of that system? This will depend upon how that component or system is to be used, i.e., how much protection is needed for the information assets that it will process?

Presumably a system where the consequences of damage to the information owning entity are catastrophic merits software that is more trustworthy with respect to integrity, availability and confidentiality than a system where the value of the information is low. Frameworks to provide guidance to developers and confidence to customers that the software or system can be specified so as to reduce the risk to protected assets and then implemented as an accurate refinement of that specification have been developed over the years, e.g. the TCSEC [1], CTCPEC [12], and ITSEC [11]. The system currently in use is the Common Criteria (CC) [23].

The CC philosophy is that assurance is increased as the level of effort applied to the security evaluation of the target of evaluation (TOE) is increased and that the level of evaluation effort can be described in seven distinct Evaluation Assurance Levels (EALs). Each EAL defines a minimum level of effort that must be applied to the IT product. The level of effort is based upon its scope, i.e. the portion of the IT product to be evaluated; depth, corresponding to the level of detail examined for both the design and implementation; and rigor, the extent to which structure and formality are applied to the development process. The seven EALs address system vulnerabilities in three ways: elimination, minimization of level of impact of the vulnerability, or through monitoring so that exploitation of the vulnerability in the operational system can be thwarted.

The CC describes ten types of assessment conducted during the evaluation of a system.[23] Each is intended to contribute to the determination of whether or not it is possible to assert that the operational system can be considered trustworthy.

1. analysis and checking of process(es) and procedure(s);
2. checking that process(es) and procedure(s) are being applied;
3. analysis of the correspondence between TOE design representations;
4. analysis of the TOE design representation against the requirements;

5. verification of proofs;
6. analysis of guidance documents;
7. analysis of functional tests developed and the results provided;
8. independent functional testing;
9. analysis for vulnerabilities (including flaw hypothesis);
10. penetration testing.

These assessment areas map to the Security Assurance Requirements (SARs) imposed on the development of the system [3] and are described in Table 2. The third and fourth columns show how our educational programs map to these requirements and will be discussed in subsequent sections.

Table 2. Common Criteria Assurance Class to Educational Component Mapping

Assurance Class	***Family***	***Class***	***TCX***
		Mapping	Mapping
Configuration Mangement	CM automation	SS	X
	CM capabilities	SS	X
	CM scope	SS	X
Delivery and Operation	Delivery	D	X
	Intallation, generation and start-up	SS	X
Development	Functional specification	SS	X
	High-level design	SS	X
	Implementation representation	SS	X
	TSF internals	SS	X
	Representation correspondence	FM, SS	X
	Security policy modeling	FM, SS	X
Guidance Documents	Administrator guidance	SS	X
	User guidance	SS	X
Life Cycle Support	Development Security	SS	X
	Flaw Remediation	SS	X
	Life cycle definition	SS	X
	Tools and techniques	FM, SS	X
Tests	Coverage	SS	X
	Depth	SS	X
	Functional Tests	SS	X
	Independent Testing	D	X
Vulnerability Assessment	Covert channel analysis	FM, SS, AVA	X
	Misuse	AVA	X
	Strength of TOE security functions *	D	—
	Vulnerability analysis	AVA	X
* This category generally applies to cryptography and is not covered in our classes			
Classes: FM - Formal Methods; SS - Secure Systems; AVA - Advanced Vulnerability Assessment			
This table is based upon Common Criteria Version 2.2			

Which EAL should be used to provide information asset owners an appropriate level of con-

fidence that their assets are protected? Given a rich lattice of damage equivalence classes from Section 1.2, and, to continue with our CC example, it is necessary to map them to the Evaluation Assurance Levels. A trivial example is a system where catastrophic consequences arise in all three areas. For this case, $\{CD_C, CD_I, CD_A\}$ could map to EAL7 and similarly $\{LD_C, LD_I, LD_A\}$ might be mapped to EAL3. But what about all of the classes in between? If we have three levels of damage: catastrophic (C), moderate (M), and low (L), does $\{MD_C, CD_I, LD_A\}$ map to the same level of assurance as $\{MD_C, MD_I, MD_A\}$? Readers can, of course, map these classes to some other software confidence scale, but the point is that is up to the information asset owner who will ultimately use the software to determine the importance of each factor. Then it is necessary to determine how those combined factors map to assurance.

In the face of a potential combinatorial explosion, educators should steer clear of endless minutia and focus on fundamental principles and technologies. One approach is to teach the students how to construct the best possible system. Then, with the knowledge that something less is often a reasonable and prudent choice for a given context, they can devlop an assurance approach to satisfy the particular requirements of the information asset owner.

2 Teaching Assurance Techniques

In this section an approach to incorporating the assurance techniques identified in the Common Criteria into a set of classes is described, the limitations of current classes are noted, and a project to remedy deficiencies created by the lack of a real-world high assurance worked example is described.

2.1 Classes

At our university a survey class on Information Assurance includes a one-week unit that provides an overview of how the Common Criteria framework can be used to evaluate the security properties of components. A second class on the Secure Management of Systems extends the overview by familiarizing students with a number of government policies mandating the use of products that have undergone CC evaluations. Although a good start, these classes do not give students a great deal of practice in the ten assurance areas identified in Table 2. The three advanced classes described below fill this need: Security Policies, Models and Formal Methods; Secure Systems; and Advanced Vulnerability Analysis.

Security Policies, Models and Formal Methods has been offered for nine years and covers the application of mathematical techniques to formulating and proving security properties, as evidenced by the course description.

> *This course covers the methods used to specify, model and verify the access control mechanisms of computational systems. The identification of the security policy and its interpretation in terms of a technical policy is covered. Several security policy and access-control models are explored.*

While reading a large number of classic papers, e.g., [13, 16, 15, 10], students learn about verification systems and complete laboratory exercises using PVS [19]. This class provides concrete experience in several of CC developmental assurance areas listed in Table 2.

Secure Systems has been offered by our department for eight years and covers

> *the implementation of protection for monolithic and distributed secure computer systems. The problems of subversion and confinement in highly secure components are*

addressed by lifecycle assurance methodologies. Topics include: protection hardware, the implementation of virtualization, synchronization mechanisms, critical sections, SWE methodologies, and configuration management techniques.

An extensive reading list covering areas ranging from hardware through software development techniques introduces students to the holistic nature of the principles and activities associated in secure system development. Initially the course attempted to address all of the Common Criteria assurance classes, except for those covered by the formal methods class. Protection of classified information adds a dimension to the class that is of considerable interest to the students and the sponsors of the curriculum.

Table 3 lists topics typically covered along with selected papers recommended as supporting material. It should be noted that the students use the Intel hardware documentation as a reference, not reading. A number of the topics depend heavily upon the personal experience of the author. In particular, the section on trusted mechanisms, which addresses guards, downgraders, and cryptographic implementations is intended to be practical and, as does the section on distributed system security, relies on a large dose of practical secure systems engineering.

Laboratory exercises have always been a critical element of the pedagogical approach [21]. All laboratory exercises involve teams of from four to six students. For three years, the laboratory exercise consisted of a penetration study of an existing system, e.g. Windows NT, Windows CE, and Linux, using the Flaw Hypothesis Methodology [29, 39]. The advantage of this kind of laboratory exercise was that students could appreciate the value of secure development methodologies. Its disadvantage was that students did not gain hands-on familiarity with such methodologies. To address this deficiency the laboratory exercise was changed to require students to develop a small component using a high assurance paradigm [22]. A new course in Advanced Vulnerability Assessment covers material on penetration testing and allows the secure system course to focus on constructive techniques for security. This has been highly effective, however, it is a very demanding and students commented that they felt that the course contained too much material. In the fall quarter of 2005, we experimented with an exercise on requirements elicitation for a highly trusted network. Although the students appeared to enjoy the project, it did not immerse them as fully in high assurance development. As a consequence, we plan to go back to the high assurance development project. The Common Criteria assurance requirements addressed by the combination of lectures and labs appear in Table 2.

Advanced Vulnerability Assessment has been offered as an experimental course for two years and this winter achieved an official catalog entry. It is focused on remote attack of target systems using tools for target assessment and explotation. Extensive labs provide practical experience with current network attack and vulnerability assessment tools as well as development of new tools. Prerequisites include Operating Systems, Introduction to Information Assurance, Network Security and Compilers. Only US citizens are permitted to enroll.

The focus of the course is not network disruption, but how to "own" the target system, i.e. how to obtain privileged user status (root) or to penetrate and control the operating system itself. Thus, students obtain considerable appreciation of how poorly developed code can be exploited for pernicious purposes. The course involves extensive laboratory work. In lieu of formal penetration testing of a specific system component, they engage in penetration testing of the composed system: applications, libraries, networking interfaces, and the operating system. This holistic approach allows them to experiment with beaded attacks in a way not possible when only a single component is examined. It addresses a number of the Vulnerability Assessment requirements of the CC (see Table 2) and relieves the Secure Systems course of the need to conduct a penetration testing exercise

Table 3. Typical Secure Systems Topics

Topic	*Readings*	*Objective*
Motivation for Trustworthy Systems	[6, 24, 25, 5]	Appreciate the security problem, its constancy across massive technological advances, and the need for a systematic approach to its solution.
Definitions and Fundamental Principles	[27, 6, 35]	Learn the vocabulary for reasoning about system security. Learn sound security engineering principles.
Requirements and Specification	[37, 3]	Study how to translate customer requirements into technical policies for a system that must also support a set of functional requirements.
Development Techniques and Evaluation	[34, 3]	Be able to apply well understood tools and techniques used to construct secure systems. Understand the secure system lifecycle discipline.
Hardware Support	[17, 18, 36, 38]	Understand fundamental mechanisms required for protection and the importance of hardware support. Appreciate the assurance and performance benefits of high quality hardware.
Structure	[14]	Learn the use of perspective in the application of principles of secure system construction and analysis.
Covert Channels and Formal Methods	[28, 26, 8, 31, 40]	Be able to state why and where covert channels are important, how they arise, techniques to avoid them, and constructive system design techniques that support covert channel analysis.
Trusted Mechanisms		Appreciate how certain mechanisms must be perform trusted functions. Learn their relationship to underlying enforcement mechanisms and how their minimization in an overall system architecture.
Distributed System Security	[32]	Be able to extend architectural concepts beyond individual systems to either logically or physically distributed architectures.

in its laboratory component.

Despite the coverage achieved by these courses, all three lack a real-world example of a inspectable high assurance system that demonstrates how systems are built from the outset to be free of unspecified functionality. The Trusted Computing Exemplar Project provides a solution.

2.2 Trusted Computing Exemplar Project

One objective of the TCX project to provide a worked example of how trusted computing systems and components can be constructed. It is anticipated that this will foster a greater appreciation of high assurance systems among educators and organizations engaged in system development. The project encompasses four complementary activities:

- Creation of a reusable high assurance development framework;
- Development of a reference-implementation trusted component;
- Support for evaluation of the reference component against EAL 7 of the Common Criteria;

- Open distribution of the results of the project results.

Each of these activities is described in greater detail.

2.2.1 High Assurance Development Framework

The TCX high assurance development framework consists of a life cycle framework and a rapid development environment. The life cycle framework, which employs an augmented spiral life cycle model, includes rigorous configuration management and developmental checks and balances that support the integrity of the effort.

The TCX development environment consists of a documentation integration environment with which to construct and manage TCX project documents; development tools and procedures for construction of TCX software; and verification tools and procedures. These developmental elements combine to allow both confidence and determination that the system defined is the system constructed. In particular, the objective of demonstrating the absence of unspecified functionality will be met to a very high degree of confidence.

2.2.2 High Assurance Reference Component

A high assurance separation kernel upon which will be layered a trusted demonstration application will constitute the high assurance reference implementation for the project.

A fundamental requirement of the TCX Separation Kernel is its a priori assurance against system subversion. It will enforce process and data-domain separation, while providing basic operating system services sufficient to support specialized applications. The kernel will have a static runtime resource configuration and its security policy regarding access to resources will be based on static process-to-resource access bindings. Scheduling will be static and a small number of processes, data objects, and I/O devices will be supported. Formal methods will be employed to demonstrate that once the kernel is brought to its initial secure state, all subsequent transitions will maintain secure state.

To further our educational objectives, we intend to develop a high assurance application to run on the TCX separation kernel. This application is a trusted path extension (TPE) that will be used to communicate security-critical information between itself and a trusted server. The TPE will be designed to support both trusted and untrusted processes, thus illustrating an often elusive architectural concepts integral to high assurance security engineering.

2.2.3 Reference Component Evaluation

The evaluation target for the TCX security kernel is EAL7, however, a validated EAL7 protection profile for separation kernels is not currently available. We are part of a team developing an EAL6+ separation kernel protection profile, for which a draft is currently available [4]. This effort has made us better prepared to develop the Security Target specification for the TCX separation kernel. Although designed and developed to meet EAL6 requirements, there are currently no plans to submit the Trusted Path Extension for formal evaluation.

2.2.4 Open Dissemination

A major objective of the project is to provide an example of how to build a high assurance trusted system. This example is intended to help educators by providing a worked example that includes various high assurance internal engineering specifications, evaluation and development framework documents, as well as code and binaries for the resulting system. The disseminated exemplar

will include, for example, all materials produced during product definition, e.g., the configuration management plan, the configuration management procedures, development standards, etc. Other phases of the project will be reflected in the documentation required for an EAL7 evaluation.

2.2.5 TCX to Assurance Class Mapping

Table 2 shows how the TCX effort will map to the assurance requirements of Common Criteria Version 2.2. It is worth noting that the Separation Kernel Protection Profile currently under development includes a number of explicit requirements that do not appear in CC Version 2.2. Several of these requirements, such as least privilege, are already reflected in our classes.

3 Summary

We have inspected the term "secure code" and have provided an argument that "secure" is judged by the information asset owner who must employ the code in order to use the asset. The concepts of *trusted* and *trustworthiness* are explored in the context of the term *secure*. This has led to the formulation of damage levels in the areas of integrity, availability and confidentiality, which pointed to criteria for assuring the trustworthiness of code and other system artifacts. As an example, a mapping of three university-level courses to assurance requirements of the Common Criteria was provided. Further, a project to provide students with a worked example of secure code development is provided.

Future work will address how to better present concepts relating to the creation of trustworthy systems. This will be achieved by weaving the results of the TCX project into our classes.

Certain entities engaged in the development of critical systems have requested some of the documentation related to the TCX rapid high assurance development framework. In response, high level requirements for the dissemination system have been articulated [33] and the dissemination system is under development.

Acknowledgment

The Trusted Computing Exemplar Project has been sponsored by the Office of Naval Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessary reflect the views of the Office of Naval Research.

References

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

[2] *Webster's Ninth New Collegiate Dictionary*. Merriam-Webster, Inc, Springfield, MA, 1986.

[3] *ISO/IEC 15408 - Common Criteria for Information Technology Security Evaluation*. Number CCIMB-2004-01-001, CCIMB-2004-01-002, CCIMB-2004-01-003. International Organization for Standardisation, version 2.2 edition, January 2004.

[4] National Security Agency. U.S. Government protection profile for separation kernels in environments requiring high robustness. Technical report, National Security Agency, July 2004.

[5] Emory A. Anderson, Cynthia E. Irvine, and Roger R. Schell. Subversion as a threat in information warfare. *Journal of Information Warfare*, 3(2):52–65, 2004.

[6] James P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA, 1972. (Also available as Vol. I,DITCAD-758206. Vol. II, DITCAD-772806).

[7] Terry V. Benzel, Cynthia E. Irvine, Timothy E. Levin, Ganesha Bhaskara, Thuy D. Nguyen, and Paul C. Clark. Design principles for security. ISI-TR-605, Information Sciences Institute, Santa Monica, California, and NPS-CS-05-010, Naval Postgraduate School, Monterey, California, 2005.

[8] Terry Vickers Benzel. Analysis of a Kernel Verification. In *Proceedings of the 1984 IEEE Symposium on Security and Privacy*, pages 125–131, Oakland, CA, April 1984. IEEE Computer Society Press.

[9] Garrett Birkhoff. *Lattice Theory*, volume 25 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 1940.

[10] David F. C. Brewer and Michael J. Nash. The chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 219–230, Oakland, 1989. IEEE Computer Society Press.

[11] Canadian System Security Centre. *Canadian Trusted Computer Product Evaluation Criteria*, Ottawa, Canada, January 1993. Communications Security Establishment, Government of Canada.

[12] Commission of the European Communities. *Information Technology Security Evaluation Criteria*, Luxembourg, 1991. Office for Official Publications of the European Communities.

[13] Dorothy E. Denning. A Lattice Model of Secure Information Flow. *Communications of the A.C.M.*, 19(5):236–243, 1976.

[14] Edsger W. Dijkstra. The Structure of the "THE"-Multiprograming System. *Communications of the A.C.M.*, 11(5):341–346, 1968.

[15] J. Goguen and J. Meseguer. Security policies and security models. In *Proceedings 1982 IEEE Symposium on Security and Privacy*, pages 11–20, Oakland, CA, 1982. IEEE Computer Society Press.

[16] M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *Communications of the A.C.M.*, 19(8):461–471, 1976.

[17] Intel. *Intel Architecture Software Devloper's Manual, Volume 1: Basic Architecture*. Intel Corporation, Santa Clara, CA, 1999.

[18] Intel. *Intel Architecture Software Devloper's Manual, Volume 2: Instruction Set Reference*. Intel Corporation, Santa Clara, CA, 1999.

[19] SRI International. PVS specification and verification system.

[20] Cynthia Irvine and Timothy Levin. *Integrity, Internal Control and Security in Information Systems*, chapter A Cautionalry Note Regarding the Data Integrity Capacity of Certain Secure Systems, pages 3–25. Kluwer Academic Publishers, Norwell, MA, 2002.

[21] Cynthia E. Irvine. Amplifying security education in the laboratory. In L. Yngstrom, editor, *Proceedings First World Conference on Information Security Education*, Stockholm, Sweden, June 1999.

[22] Cynthia E. Irvine. A common criteria-based team project for high assurance system education. In *Success Through Information Security Knowledge, Proc. Fourth World Conference on Information Security Education*, pages 82–93, Moscow, Russia, May 2005.

[23] ISO/IEC. ISO/IEC 15408 - Common Criteria for Information Technology Security Evaluation. Version 3.0, July 2005.

[24] Paul A. Karger and Roger R. Schell. Multics security evaluation: Vulnerability analysis. Technical Report ESD-TR-74-193, Vol. II, Information Systems Technology Application Office Deputy for Command and Management Systems Electronic Systems Division (AFSC), Hanscom AFB, Bedford, MA 01730, 1974.

[25] Paul A. Karger and Roger R. Schell. Thirty years later: The lessons from the multics security evaluation. In *Proceedings 18th Annual Computer Security Applications Conference*, pages 119–126, Las Vegas, December 2002. IEEE Computer Society.

[26] Richard A. Kemmerer. A Practical Approach to Identifying Storage and Timing Channels. In *Proceedings 1982 IEEE Symposium on Security and Privacy*, pages 66–73, Oakland, CA, April 1982. IEEE Computer Society Press.

[27] Butler W. Lampson. Protection. In *Proc. 5th Princeton Conf. on Information Sciences and Systems*, pages 437–443, Princeton, NJ, 1971.

[28] Butler W. Lampson. A note on the confinement problem. *Communications of the A.C.M.*, 16(10):613–615, 1973.

[29] Richard R. Linde. Operating system penetration. In *National Computer Conference*, pages 361–367, 1975.

[30] Steven B. Lipner. Non-discretionary controls for commercial applications. In *Proceedings 1982 IEEE Symposium on Security and Privacy*, pages 2–20, Oakland, 1982. IEEE Computer Society Press.

[31] Jonathan Millen. Covert channel capacity. In *Proceedings 1987 IEEE Symposium on Security and Privacy*, pages 60–66, Oakland, 1987. IEEE Computer Society Press.

[32] Jonathan Millen. The cascading problem for interconnected networks. In *Fourth Aerospace Computer Security Applications Conference*, pages 269–273, 1988.

[33] Thuy D. Nguyen, Cynthia E. Irvine, and Douglas R. Kane. Using common criteria methodology to express informal security requirements. In *Proceedings International Symposium on Secure Software Engineering*, page to appear, Arlington, VA, 2006.

[34] David L. Parnas. On the critera to be used in decomposing systems into modules. *Communications of the A.C.M.*, 15(12):1053–1058, 1972.

[35] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[36] Olin Sibert, Phillip A. Porras, and Robert Lindell. The Intel 80x86 processor architecture: Pitfalls for secure systems. In *Proceedings 1995 IEEE Symposium on Security and Privacy*, pages 211–222, Oakland, CA, May 1995. IEEE Computer Society Press.

[37] Daniel F. Sterne. On the buzzword "security policy". In *Proceedings of the IEEE Symposium on Research on Security and Privacy*, pages 219–230, Oakland, CA, May 1991. IEEE Computer Society Press.

[38] Trusted Computing Group. TCG specification architecture overview. Technical Report Rev 1.2, Trusted Computing Group, 28 April 2004.

[39] Clark Weissman. *Security Penetration Testing Guideline*. Naval Research Laboratory, Unisis Government Systems, 12010 Sunrise Valley Drive, Reston, VA, tm-8889/000/01 edition, October 1993. prepared undercontract to NRL. Available on the WWW.

[40] J.C. Wray. An Analysis of Covert Timing Channels. In *Proceeding of the IEEE Symposium on Research in Security and Privacy*, pages 2–7, Oakland, CA, May 1991. IEEE Computer Society Press.

[41] Che-Fn Yu and Virgil D. Gligor. A formal specification and verification method for the prevention of denial of service. In *Proceedings 1988 IEEE Symposium on Security and Privacy*, pages 187–202, Oakland, CA, April 1988.