A Common Criteria-Based Team Project for High Assurance Secure Systems

Cynthia E. Irvine
Department of Computer Science
Naval Postgraduate School, Monterey, California, USA

Abstract

Most courses in information security do not provide students with practical experience in high assurance development. To complement a course in secure systems that focuses on foundational principles of constructive security, a laboratory project that requires students to work in teams while meeting Common Criteria Evaluation Assurance Level (EAL) 6 assurance requirements has been created. The objectives, structure, and experience with this laboratory project are described.

Key Words: Information Assurance, Software and System Assurance, Common Criteria, Laboratory Project

1. Introduction

Introductory computer security courses often review such a broad range of concepts and mechanisms that students are unable to delve into the subtleties of secure system development. Even when advanced courses in computer security are offered, lectures and readings can eclipse laboratory work and projects. We have developed a course in secure systems that covers the foundational principles of high assurance secure system development. Although the course encompasses a very substantial amount of lecture and reading material regarding secure system development, laboratory activity is considered to be an essential to the learning process. To reinforce the ideas presented in the abstract and to give students a small amount of concrete experience in secure system development, we have created a unique Common Criteria-based laboratory project.

This paper is intended to describe the context of the advanced class in secure systems as well as its current laboratory activities. To this end, the history of laboratory work in the class will be reviewed and then the objectives of and observations regarding the current project will be presented.

1.1 Context of the Course

The Computer Science Department at the Naval Postgraduate School offers a broadly-based program in information and software assurance. The current set of classes offered in system security and assurance are described in Table 1. Of this set, the first course, entitled Introduction to Information Assurance: Computer Security, is both a prerequisite to all subsequent offerings as well as a requirement for all students campus-wide in programs related to the information sciences and information technology. The university is on the quarter system and each quarter lasts eleven weeks, thus all courses are eleven weeks in duration.

Table 1. Courses in Information and Software Assurance

Course Title	Hours*	Description
Introduction to Information Assurance: Computer Security *(core requirement)*	4-2	This is a comprehensive overview of the terminology, concepts, issues, policies, and technologies associated with the fields of Information and Software Assurance. It is a prerequisite for all other Computer Security Track courses.
Analysis of Government Critical Infrastructure Protection	3-1	A government-formulated CIP lifecycle is introduced, that includes: Analysis & Assessment, Remediation, Indicators & Warnings, Mitigation, Incident Response, and Reconstitution. Assessment of the criticality and survivability of mission-critical infrastructures is covered.
Information Assurance: Secure Management of Systems *(track requirement)*	3-2	A security manager's view of the diverse management concerns associated with administering and operating an automated information system facility with minimized risk is presented. Students examine both the technical and non-technical security issues associated with managing a computer facility.
Vulnerability Assessment	3-2	The potential vulnerabilities in networked systems are taught by studying methods to: obtain information about a remote network and to possibly exploit or subvert systems on that network. Labs provide practical experience with current network attack and vulnerability assessment tools.
Network Security *(track requirement)*	4-2	This course covers the concepts and technologies used to achieve confidentiality, integrity, and authenticity for information processed across networks.
Secure Systems *(track requirement)*	3-2	The implementation of protection for individual and networked secure computer systems is covered. The problems of subversion and confinement in highly secure components are addressed by coherent mechanisms and lifecycle assurance methodologies.
Database Security	3-1	Course topics include: policies for information integrity and confidentiality of database (DB) systems, modeling of secure DB systems, security in statistical and object-oriented DBs, multi-tier architecture security issues, privacy, aggregation and inference, and applications of secure DBs.
Security Policies, Models, and Formal Methods *(track requirement)*	3-1	The methods used to specify, model and verify the access control mechanisms of computational systems are presented. 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.
Mobile and Wireless Security	3	This course addresses the security functionality, protocol and assurance issues associated with this emerging technology.
Computer Forensics	3-2	This course covers the fundamentals of computer forensics in the context of government information operations. Students examine how information is stored and how it may be deliberately hidden and/or subverted.
Introduction to Certification and	3-2	This course provides an introduction to the Certification and Accreditation (C&A) process as applied to the procurement and

Course Title	Hours*	Description
Accreditation		lifecycle management of government information systems.
Advanced Topics in Computer Security	3-1	Advanced topics in software, communications, and information security are covered. Information security policies are studied, along with issues involving existing and emerging IA issues and technologies.

* Hours should be read as follows: n lecture hours – m laboratory hours per week

All students must take a set of core computer science classes, which are augmented by specialization, or "track", requirements. The security track mandates the following core courses: Introduction to Information Assurance: Computer Security; Information Assurance: Secure Management of Systems; Network Security; Secure Systems; and Security Policies, Models and Formal Methods. Each student can then select three electives to complete their course of study. Certain scholarship programs supported by the Security Track require additional courses, such as Database Security and Advanced Topics in Computer Security.

Our university requires that every Masters student write a thesis based upon research conducted during the final year of his or her studies. Each thesis is supervised by one or more members of the faculty and affords students an opportunity to participate in one of the many ongoing research projects within the department. Often students will select electives that enhance their ability to conduct the thesis research. Often students will enroll in courses outside of the security track and even outside of the department.

Doctoral students are not required to take a specific set of track courses, but must pass a set of three qualifying examinations: one on the formal foundations of computer science, and two others chosen from among the specializations of the department. One of these is an examination in information assurance. All doctoral students take, as a minimum, Network Security; Secure Systems; and Security Policies, Models and Formal Methods.

From the preceding discussion, it clear that Secure Systems is a key course for both Masters and Ph.D. students [14]. The course has been taught for over seven years at our university and has evolved over that period. Its principled-based foundations remain constant, however its laboratory activities have evolved.

1.2 Previous Secure Systems Laboratory Exercises

As seen in Table 1, each security track course has both lecture and laboratory hours. Laboratory activities may involve short tutorial-style exercises, examination of case studies, weekly projects, or longitudinal projects. These hands-on activities may be used to reinforce concepts covered in class, but in some cases they are employed to convey ideas that cannot be learned through lectures alone [11]. It is the premise of the laboratory exercises for Secure Systems that there are concepts that cannot be internalized without actually doing them.

For many years the laboratory exercise taught in Secure Systems involved a Flaw Hypothesis Methodology (FHM) [15] [19] study of an existing system. Systems examined included Windows NT, Windows CE, Linux, and OpenBSD. None were chosen because they offered superior security, but rather because they were in mainstream use. Students were organized in teams where each student had both an analytical and an operational role. For the analytical roles, each student would learn as much as possible about an assigned subsystem of the operating system. The operational roles included responsibilities such as System Administrator, Flaw Hypothesis Database Manager, Programmer, Project Leader, etc.

Students were assessed in three areas: collective results of the team, analytical role and operational role. This was made possible through the project product: a Flaw Hypothesis Methodology Report. Students were required to create a single coherent report, but each student had to contribute sections describing the results of the subsystem analysis as well as how the operational role was fulfilled.

Despite the fact that the students enjoyed the project, they found it to be rather overwhelming. They were required to learn a lot about a very large operating system in a very short time, and only then were they able to conduct their flaw hypothesis study. Another cause of frustration occurred when proprietary operating systems were used. In these cases, students felt as if they were poking at a black box. When open source systems were used, the students felt compelled to study voluminous source code listings. Note that a professional Flaw Hypothesis team undertaking an analysis as part of an operating system evaluation would have the following: lots of experience in operating systems, experience in the FHM, full system documentation, and plenty of time. The students had little experience and little time.

The MINIX operating system was not selected as a target for the flaw hypothesis study, because it is intended a system for the study of operating system construction by students. Those chosen for the Flaw Hypothesis work were selected because they were mainstream operational systems rather than study pieces. However, MINIX is a very appropriate target for student enhancements.

Since the principle objective of Secure Systems is to teach the concepts of constructive security [12], a laboratory project based upon the construction of a small security enhancement to an existing operating system was designed. Students would use a high assurance development methodology for the enhancement and, thus, would become familiar with the checks and balances required to construct a system when one of the developmental threats is subversion. Because students were studying MINIX in a prerequisite operating systems course, it became the logical target for the high assurance security enhancements.

2. A New Secure Systems Laboratory project

To frame the project, the Reference Monitor Concept was used as the guiding principle for secure systems design. The MINIX operating system was chosen to be the target operating system for the project and Common Criteria [5] assurance requirements were used as requirements for the development process. Each will be discussed in greater detail.

2.1 Reference Monitor Framework

This section is intended to provide a brief overview of the Reference Monitor Concept as a motivating element for the course and accompanying laboratory activity.

The Reference Monitor Concept [2] is a powerful tool for defining and assessing system security properties. It has three requirements:

First, the access mediation mechanism is always invoked so that every access is mediated. If this were not the case, then it would be possible for an entity to bypass the mechanism and violate the policy that must be enforced.

Second, the access mediation mechanism is tamperproof. In the model, it is impossible for an attacker to penetrate the protection mechanism such that the required access checks are

not performed with a consequent failure to enforce the security policy allocated to the mechanism.

Third it must be understandable. Understandability permits the mechanism to be examined so that confidence is established in its ability to meet the first two requirements. In general, understandability is enhanced through the minimization of unnecessary complexity and through a careful design process.

To date, the Reference Monitor Concept is the only effective paradigm we know of for describing the abstract requirements of secure system design and implementation. No viable alternative has been introduced and it has proven effective under close scrutiny. In light of the fact that many vendors do not elect to take a constructive approach to the design and implementation of secure systems, it may be more important than ever to ensure that students understand how the constructive security guided by the reference monitor can be applied across a broad range of systems, while providing an analytical approach to systems that draws from many disciplines: programming languages, software engineering, computer engineering, operating systems, networks, etc. [10]

2.2 MINIX

It is very useful for students to examine and work with a real operating system in an operating systems course. In the early days of UNIX, it was available and used by many universities. To fill the void created when AT&T withdrew UNIX as an available, living example of operating system source code, Tannenbaum wrote a pedagogical operating system that presented a UNIX-compatible interface and made it available to the academic community [18]. It is a small, modular operating system. Over the years, Tannebaum has resisted pressures to add features to MINX and it remains a simple operating system intended as an educational tool. In contrast, LINUX has evolved into a major operating system with a vast feature set [3].

MINIX provides interface functions for process management, synchronization, file management, directory and file system management, and time management. Its protection functions allow the user to manage the protection attributes of files and directories, presenting the UNIX-like semantics of read/write/execute protection bits. The source code listing is under 28,000 lines and includes numerous in-line comments. Other than the protection bits, MINIX offers little in the way of protection features. It does use protected mode and has protection bits on files, but, for the most part, security has not been given high importance. (Less than 20 pages is devoted to security in Tannenbaum and Woodhull [18])

At our university, MINIX is used in the operating systems class that is a prerequisite to the secure systems class. This made it a natural choice for student-devised security improvements.

2.3 Common Criteria

The Common Criteria [5][6][7] is an international standard used as the basis for the evaluation of the security properties of information technology products and systems. It permits independent security assessments to be conducted on these products and systems so that consumers can have assurance that security requirements have been met. This enables consumers to select products and systems based upon unbiased evidence rather than vendor claims.

Security requirements for a Target of Evaluation (TOE) can include confidentiality, integrity and availability and are expressed in an implementation-independent Protection Profile and in an system-dependent Security Target. The functional and assurance requirements are derived from Parts 2 and 3 of the Common Criteria.

Because MINIX has not been constructed to meet a protection profile or security target and since it would be impossible in an eleven-week class to somehow transform it into a high assurance product, we focus instead upon the assurance of the security extension.

Assurance requirements are categorized into classes and families and are briefly depicted in Table 2. The first column is the name of the assurance class, and the second column is the assurance family within that class. (The interested reader is referred to Section 2.6, Part 3 of the Common Criteria for an overview of the assurance classes and families.) The third and fourth columns give the component and a brief description for Evaluation Assurance Level (EAL) 6. The requirement levied EAL 6 is given along with an indicator of the applicability of the requirement to the class project. Sometimes a requirement depends upon the security enhancement being implemented. For example, in the case of guidance documents, students are required to provide documentation only if the project results in new interfaces for the users or administrators of the system. This is indicated by P (possibly), while other requirements are indicated by Y (yes) or N (no).

Table 2. Common Criteria Assurance Classes and Families required for the laboratory project

Assurance Class	Assurance Family	Abbreviated Name and Component	Component Title	EAL6	Project
ACM: Configuration Management	CM automation	ACM_AUT.2	Complete CM automation	2	Y
	CM capabilities	ACM_CAP.5	Advanced support	5	Y
	CM scope	ACM_SCP.3	Development tools CM Coverage	3	Y
ADO: Delivery and Operation	Delivery	ADO_DEL.2	Detection of modification	2	N
	Installation, generation, and start-up	ADO_IGS.1	Installation, generation and start-up procedures	1	N
ADV: Development	Functional Specification	ADV_FSP.3	Semiformal functional specification	3	Y
	High-level design	ADV_HLD.4	Semiformal high-level explanation	4	Y
	Implementation representation	ADV_IMP.3	Structured implementation of the TSF	3	Y
	TSF Internals	ADV_INT.2	Reduction of complexity	2	Y
	Low-level design	ADV_LLD.2	Semiformal low-level design	2	Y
	Representation correspondence	ADV_RCR.2	Semiformal correspondence demonstration	2	N
	Security policy modeling	ADV_SPM.3	Formal TOE security policy model	3	N
AGD: Guidance Documents	Administrator guidance	AGD_ADM.1	Administrator guidance	1	P
	User guidance	AGD_USR.1	User guidance	1	P
ALC: Life cycle support	Development Security	ALC_DVS.2	Sufficiency of security measures	2	Y
	Flaw remediation	ALC_FLR	Not applicable (n/a)	n/a	N
	Life cycle definition	ALC_LCD.2	Standardized life-cycle model	2	Y
	Tools and Techniques	ALC_TAT.3	Compliance with implementation Standards – all parts	3	Y

Assurance Class	Assurance Family	Abbreviated Name and Component	Component Title	EAL6	Project
ATE: Tests	Coverage	ATE_COV.3	Rigorous analysis of coverage	3	Y
	Depth	ATE_DPT.2	Testing: low-level design	2	Y
	Functional Tests	ATE_FUN.2	Ordered functional testing	2	Y
	Independent Testing	ATE_IND.2	Independent testing-sample	2	N
AVA: Vulnerability Assessment	Covert channel Analysis	AVA_CCA.2	Systematic covert channel analysis	2	N
	Misuse	AVA_MSU.3	Analysis and testing for insecure states	3	N
	Strength of TOE security functions	AVA_SOF.1	Strength of TOE security function evaluation	1	N
	Vulnerability analysis	AVA_VLA.4	Highly resistance	4	N

3. Laboratory Project Objectives

So that we would be able to assess the success of the project, a set of high-level objectives was formulated.

3.1 Understand concepts of high assurance development

The assurance requirements of the Common Criteria provide the basis for the development methodology used by the students. Although students may have taken courses in software methodology and software engineering, they have never applied and augmented these processes to thwart a malicious insider intent on system subversion [1]. Using a subversion opportunity analysis [16] as guidance, students can better understand how each element in the high assurance security engineering process is both necessary and sufficient for the development of an evaluatable product.

As part of the class a number of papers pertinent to high assurance development are studied. Several are particularly relevant to the project. For example, Dijkstra's paper on the THE Operating System [8] describes the analysis required to ensure that a system is hierarchically layered and tested for correctness. The use of modularity and data hiding as described by Parnas [17] must be applied to the design and implementation of the enhancement. While developing their security enhancement, students obtain a better appreciation the concepts presented in these and other papers.

3.2 Teamwork and Leadership

Too often, university students are tasked with individual projects. In the development of real software and systems, individual genius is rarely the paradigm. Instead teams are organized to develop the product or system. Clearly, if the team consists of bright individuals, the results may be highly innovative and well constructed, but a product developed by a loner is a rarity. Unfortunately, the lack of team building projects in computer science combined with the rather introverted nature of most computer science students results in serious acculturation problems following graduation. Therefore, an objective of the project is to force the students to work together as a team. The project is large enough that it impossible for a single student, who is also enrolled in several other classes, to succeed single-handed.

Members of the team have various responsibilities. One of the more important roles is that of team leader. Because our university has students with leadership skills, team leaders emerge naturally. A good team leader can provide a role model to the less experienced students who can observe how a leader helps the members achieve their goals through project management and social skills. The team leaders are not experienced in software project management, so they learn how to harness the talents of their diverse team members to bring the effort to a successful conclusion.

3.3 Ancillary Objectives

A tangential objective of the project is that students obtain a more detailed appreciation of the structure of an operating system. Their previous experience with MINIX combined with their work on the security enhancement significantly increases their comfort level with respect to operating systems.

Another objective of the project is to reinforce the students understanding of the Common Criteria. As graduates of a security program, they may be called upon to compare systems that lack Common Criteria Evaluations with systems that have been evaluated. The experience of the Secure Systems project allows them to appreciate the requirements of the Evaluation Assurance Levels and the contexts in which security benefits gained at the higher assurance levels are applicable.

4. Exercise Organization

Careful organization is required to ensure that the projects are feasible within the academic quarter.

4.1 Schedule

Secure Systems is an eleven-week course, so an aggressive schedule is necessary to ensure that the students have sufficient time to complete their tasks. During the first week of class, teams are organized and students are familiarized with the types of documentation required for their secure development methodologies. This means that, although student teams might not have identified the specific security enhancement they want to implement for MINIX, they are able to start work on the development documents and procedures. By the third week of class, teams have identified their proposed security enhancements.

Completed projects are ready for presentation in class and peer evaluation by the end of the eighth week. Peer evaluations are completed at the end of the ninth week, thus providing each team with two weeks to implement final improvements to the project.

4.2 Expected Products

Each team is required to select an enhancement to MINIX and then create a high assurance development framework in order to implement that enhancement. A project report created by the team is the product. Each student is expected to play several roles on the team. For example, a student might serve as the team's system administrator and might also be the author of the software development standards. Both aspects of the student's contribution must appear in the report.

The report should contain descriptions of the configuration management plan, life cycle management plan, coding standards, minutes of meetings, design documentation, code, configuration control board decisions, test plans, test results, and considerable other pertinent documentation. Students are expected to make a presentation of their work in class approximately three weeks before the end of the course. At that time, their work is turned over to one of the other teams, who assume the role of evaluators.

The evaluators are expected to produce a short report on the project, copies of which are made available both to the developers of the security enhancement and to the instructor. The "evaluation report" is due two weeks before the end of the course, thus allowing each team an opportunity to make changes to their project prior to submitting it to the instructor. The evaluation report is constructed by comparing the Common Criteria requirements with the project report.

4.3 Team Composition, Assets and Motivators

Students taking the secure systems course are quite diverse. Most classes have several scholarship students, a number of young students who entered the Masters program immediately upon receiving their undergraduate degrees in computer science, and more mature students. The majority of the more mature students, who are generally between eight and ten years past receiving their undergraduate degrees, have some leadership experience. Among the more experienced students, there are usually several foreign students.

To ensure that teams do not become enclaves of one type of student, for example only scholarship students or only students for whom English is their second language, constraints are made on team formation. On the first day of class a roster is handed around that includes a few questions to be answered by each student. For example, students are asked how many years it has been since their undergraduate degree and if English is the language with which they are most comfortable.

The answers to these questions become the basis for team formation constraints, which are announced at the start of the next class. In the most recent offering of the course, each team was required to have no more than one of the very youngest students, at least one scholarship student, and at least one experienced, mature student. This balance ensured that teams had the combination of the high energy of youth and the wisdom of experience, where the latter would ensure that the team would not wait until the day before a milestone to start work on the current phase of the project.

The project requires the use of a variety of systems. Students are encouraged not to use the same platform for their target system and for their configuration management system. Most of the students in the course have their own computers. In addition, our university provides general computing support, with many applications for project management and document production.

A few years ago, our department acquired approximately twenty laptops that faculty can reserve for the use of students taking a specific class. To ensure that the students have target systems for the enhanced versions of MINIX, a minimum of one laptop per team is reserved.

As a motivator for excellence in the project, each class is informed that the team with the best overall project will be given a half a letter grade bonus. Since this course involves a number of difficult exams and other challenging assignments, the students earnestly pursue this motivator.

4.4 Student Assessment

In a team project it is important to know the roles assigned to each team member and to devise a way so that each student can be judged for his or her individual contributions to the team effort. Fortunately high assurance development calls for the creation of a number of documents as well as a number of roles. The team had to come to a consensus regarding the roles to be assumed by each student and the documents each student was responsible for writing. When the final project report is turned in, students must describe their contributions in two major areas: documents created and team responsibilities.

Grading consisted of the following elements, described in Table 3.

Table 3. Project Assessment

Project Component	Percent of Grade	Comments
Overall Project	30	The quality of the overall team effort was graded.
Documents Created	30	To create a methodology for high assurance development, each team had to develop a number of documents. Each student wrote one or more of these documents for the team.
Team Responsibilities	30	These responsibilities related to the role(s) assumed by the student and how well they were performed. The description of the activities associated in the role documented in the final report were used in the evaluation.
Team Feedback	10	Feedback from team members constituted the final portion of the assessment.

For the instructor, grading the projects has been arduous. Of course, the students have an opportunity to demonstrate the functionality their security enhancement, but since this is not the point of the exercise, the instructor faces a considerable amount of paperwork. Each project report has been over 100 pages long.

5. Discussion

Despite the challenge associated with creating a high assurance development environment in a rather short time, feedback from students has been positive. The rather short time frame makes the project very difficult, so the project might be easier at universities on a semester rather than a quarter schedule.

5.1 Results

Our objective was to instill in the students an appreciation of the processes necessary to provide assurance that subversion of their deliberately small enhancement to MINIX did not occur. Students have been able to create effective and successful development frameworks. Minimization of the scope of the security enhancements is essential for the success of the projects.

5.2 Problems

Problems are inevitable in any project-oriented undertaking. Below, several of the major difficulties encountered in the exercise are discussed.

5.3 Students Drop Course

Because each team member has specific responsibilities with respect to the project, serious problems can occur if a student drops the course. Due to the number of roles involved, a minimum acceptable team size of four is necessary. To avoid these problems, students are polled at the beginning of the class to determine if there is anyone in the class who does not want to be there. Then the teams can be organized in anticipation of a student drop.

On a few occasions, unforeseen circumstances resulted in withdrawal of a student from the class. If this draws the team size down to four, no large-scale reorganization was required; however, if the membership of the team goes below four, drastic steps are taken: each remaining team member is reassigned to another team, where small teams were chosen preferentially.

5.4 Non-contributing Team Members

Occasionally a team will have a member who is a laggard. Discussions with the instructor regarding team progress allow both the team leader and the instructor to seek ways to solve the problem. Team feedback forms allow team members to indicate team members who are not contributing adequately, but in a team effort this does not help get the job done. In such cases, the instructor and the project leader can come to an agreement that, if the laggard team member does not make the required contribution, the rest of the team would not be penalized. Fortunately, such dire circumstances have not occurred, but they clearly need to be considered when developing team projects.

5.5 Project Scope

One team proposed construction of an encrypting file system for MINIX. Although a major security improvement to the existing MINIX system, it was clearly too large: the students would not have been able to complete the project by the end of the class. The instructor must be alert to over ambitious programming projects because at the outset of the project, students are unaware of the amount of effort required to construct a high assurance development environment. Had the objective of the course been extensive practice in system-level design and implementation, then a large project might have been appropriate. In this case the process was deemed more important than what was to be built.

5.6 Comparison with Other Work

Du [9] has developed a version of MINIX called SMINIX intended for use in teaching security mechanisms in operating systems. MINIX was enhanced by the faculty to include the following security features: authentication, access control lists, label-based access control, capabilities, sandboxing, a Secure file system, and privilege mechanisms. Once SMINIX was constructed, the interfaces to the new security features were left in place, but the mechanisms were turned into stubs. In a sequence of laboratory activities of increasing difficulty, students must implement the mechanism while remaining faithful to the module interface.

Although this approach teaches students details of the security mechanisms, they do not learn about the development processes required to achieve assurance. In addition, teamwork is not emphasized; the projects can be completed individual students.

Nachos [4] is a popular framework for providing experience in operating system development, however, it does not emphasize security and its use does not necessarily result in practical experience in either assurance or security mechanisms.

Our project is intended to teach the process of high assurance software and system development in the context of an international standard for secure system evaluation, rather than the particulars of specific security mechanisms. Both SMINIX and Nachos could be adapted to our teaching objectives. Combined with its emphasis on teamwork, these factors differentiate our laboratory project from the usual system enhancement activities used in many operating system classes.

5.7 Future Work

Our research group is currently in the process of developing a high assurance example of trusted computing [13]. The project consists of four related activities: creation of a framework for the rapid development of high assurance systems, development of a reference-implementation trusted computing component; evaluation of the component for high assurance; and open dissemination of the results related to the first three activities.

As this project progresses, we intend to use the reference-implementation trusted computing component as the basis for the Secure Systems laboratory work instead of MINIX. Then, instead of making improvements to a patently insecure, low assurance system, the students will be charged with the development of an enhancement to an existing high assurance system. The scope of the programming effort is expected to increase along with a change in the balance between the creation of procedures and standards, and design and implementation evidence.

In summary, this project has been useful in that it gives students practical insight into the complexity of the high assurance development process and allows them to work as a team to develop a rudimentary, but workable, high assurance development framework.

Acknowledgments

This work has been possible through the participation of the many students who enrolled in Secure Systems. The comments of an anonymous referee were helpful in the creation of the final version of this paper.

References

[1] Anderson, Emory A., Irvine, Cynthia E., and Schell, Roger R., "Subversion as a threat in information warfare," *Journal of Information Warfare*, Vol. 3, No. 2, pp 52-65.

[2] Anderson, J. P., 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 DTCAD-772806).

[3] Bovet, D. P., and Cesati, M., *Understanding the LINUX Kernel*, O'Reilly, Sebastopol, CA, 2001.

[4] Christopher, C., Procter, S., and Anderson, T., "The Nachos instructional operating system," Proc. 1993 Winter USENIX Conference, January 1993, pp. 479-488.

[5] Common Criteria for Information Technology Security Evaluation, Version 2.2, CCIMB-2004-01-00[1,2,3], January 2004.

[6] Common Evaluation Methodology for Information Technology Security, Part 1: Introduction and general model, CEM-97/017, Version 1.0, August 1999.

[7] Common Evaluation Methodology for Information Technology Security, Part 2: Evaluation Methodology, CEM-99/045, Version 0.6. January 1997.

[8] Dijkstra, E., The Structure of "THE"-Multiprogramming System, *Communication of the ACM*, Vol. 11, No. 5, May 1968, pp. 345-346.

[9] Du, W., "Developing an instructional operating system for computer security education," *in 7th Colloquium for Information Systems Security Education*, Washington, DC, June 2003.

[10] Irvine, Cynthia E., "The reference monitor concept as a unifying principle in computer security education," *Proceedings of the First World Conference on Information Systems Security Education*, Stockholm, Sweden, pp.27-37, June 1999.

[11] Irvine, Cynthia E., "Amplifying security education in the laboratory," *Proceedings of the First World Conference on Information Systems Security Education*, Stockholm, Sweden, pp. 139-146, June 1999.

[12] Irvine, Cynthia E., "Teaching constructive security," *IEEE Security and Privacy*, Vol. 1, No. 6, pp 59-61, 2003.

[13] Irvine, Cynthia E., Levin, Timothy E., Nguyen, Thuy D., and Dinolt, George W., "The Trusted Computing Exemplar Project," *Proceedings of the 2004 IEEE Systems Man and Cybernetics Information Assurance Workshop*, West Point, NY, June 2004, pp. 109-115.

[14] Irvine, Cynthia E. and Levin, Timothy E., "A doctoral program with specialization in information security: A High Assurance Constructive Security Approach," *Information Security Management, Education and Privacy*, ed. Deswarte, Cuppens, Jajodia, and Wang, Kluwer Academic Publishers, Norwell, MA, pp. 173-180, 2004.

[15] Linde, "Operating system penetration", *in National Computer Conference*, pages 361-368, 1975.

[16] Myers, P., "Subversion: the neglected aspect of computer security," Masters Thesis, Naval Postgraduate School, 1980.

[17] Parnas, D. L., "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, Vol. 15, No. 12, December 1972 pp. 1053 – 1058.

[18] Tannenbaum, A., and Woodhull, A.S., *Operating Systems: Design and Implementation*, 2nd Edition, Prentice Hall, Upper Saddle River, NJ, 1997.

[19] Weissman, "Penetration testing," *in Handbook of Computer Security Certification of Trusted Systems*, Technical report, Naval Research Laboratory, January 1995. NRL Technical Memorandum 5540:082A.