Amplifying Security Education in the Laboratory

Cynthia E. Irvine
Naval Postgraduate School, Monterey, CA 93943 USA

Key words: Computer Security Education, Laboratory Exercises

Abstract: Computer and network security have become concerns for enterprises ranging from sole proprietorships run from home offices to global corporations and government agencies with hundred of thousands of employees. These concerns are reflected in the growing demand for computer security professionals to design, manage, and administer systems. Here a case is built for significant use of laboratory work to complement classroom and reading activities in computer security education.

1. INTRODUCTION

Computer science and computer engineering are young fields. Each is still evolving as principles are formulated and new technologies developed. As sub-disciplines emerge, those conducting college- and university-based research and education examine how best to revise curricula. The objective is to maintain an environment that creates graduates with the essential skills and knowledge needed to participate in a information technology-oriented society. Inventions and new ideas will expand our horizons, yet they will be grounded in the theoretical and technical foundations of our science. It is this common base that permits cross-fertilization and continued growth.

In many respects, the maturing of software engineering as an academic discipline may provide a template for the processes we are now just initiating in computer security education. The importance of software engineering as a discipline was recognized as poor design, incomplete specifications, lack of extensibility, ill-defined or non-existent configuration management, and mistakes resulted in ballooning software costs. Study of the software engineering process has permitted articulation of principles that not only guide future research, but lead to practical approaches to the construction of complex systems. Yet, we know that advances in software methodology have been made possible by progress in many areas. Programming languages and compilers help with strong type checking. Advances in hardware have relieved us of many of the time-memory tradeoffs that tormented early system developers.

In computer and network security we face challenges that, in many ways, are even more daunting than those encountered by software engineers. While software engineers are interested in constructing robust, maintainable, extensible, validated and verified code, security professionals must go beyond those requirements to design, construct, and maintain systems that must support the enforcement of security policy in the face of abuse and misuse. These include: user carelessness and unwitting errors; white collar crime; system interface abuse; automated attacks on the interface; exploitation of system flaws; and subversion of critical components during the system lifecycle [1]. Security mechanisms must work while adversaries are actively attacking them.

Computer security is not a discipline for the isolationist. We must understand and bring to our problems the best that computer science and computer engineering have to offer. We must use the ideas, tools and technologies of our colleagues in software engineering, programming languages, user interface design, hardware, networking, operating systems, etc. to construct systems. Ultimately, governments, enterprises, small businesses and individuals must be able to make a technical judgement that the systems we produce are adequate for the enforcement of their security policies, ranging from those that protect national secrets to privacy and integrity controls for personal banking files on home computers. What we bring to computer science is our knowledge of the theory and principles that underlie computer security; an understanding of the science as a whole and of its new technologies; and the creativity to invent new ways to build systems that will encode specified security properties.

From this perspective, it is clear that computer security education must produce individuals who have a broad understanding of the scope of the discipline as well as considerable knowledge and expertise in specific areas. The question to be addressed here is how laboratory work can complement classroom study. First, it is important to decide whether laboratory activities will benefit the educational process. Next, we need to examine whether there are laboratory activities unique to computer security education. If the answer to the previous query is affirmative, then the particular objectives of our educational programs must be examined and appropriate laboratory exercises selected. A few examples will illustrate this last point.

2. ARE LABORATORY EXERCISES NEEDED?

Before we can discuss how laboratory exercises amplify the educational process in computer security, it is necessary to establish that exercises are desirable. Computer science is not pure mathematics, nor is it pure engineering. Instead it is a happy marriage of the two. It is mathematics, in the form of computability, algorithms, models, etc., that must relate to a concrete reality, and engineering that must use mathematics to both ensure against attempts to construct impossible systems and allow us to build working, understandable systems. We can look at worked examples where our colleagues in other sub-disciplines of computer science have used laboratory work to complement the classroom. Certainly there are courses in which laboratory work, although possible, is often not necessary. Examples are: automata, algorithms, programming language theory, and complexity theory. However, few would dispute the benefit of laboratory work in, for example: operating systems, networks, compilers, databases, software engineering, graphics and visualization. Indeed, who would even contemplate teaching students a particular programming language without programming exercises in the laboratory?

3. UNIQUE TOPICS IN COMPUTER SECURITY

With all of the laboratory activities that abound in traditional computer science, is there a need for specific laboratory work in computer security education? What kinds of laboratory exercises are unique to this discipline? Curricula in computer security contain unique topics, many of which lend themselves to laboratory work. A possible classification scheme for laboratory exercises contains five broad areas. They are:

1. Hands-on studies of system vulnerabilities and exercises in penetration analysis;
2. Experiments using various security test and enhancement products;
3. Exercises to enhance student understanding of security concepts and principles;
4. Work to familiarize students with tools used in the construction of secure systems; and
5. Projects to construct secure systems or subsystems.

Will exercises in any of the above areas be conducted as part of "standard" computer science education? In curricula that consciously attempt to integrate computer security into the general program, it is possible that security may be a theme for laboratory activities [4, 2]. Certainly, a course in software methodology contributes topics in the use of tools and robust engineering practice, however, to succeed, this work must address the concern for intentional malice that characterizes computer security. Unfortunately, topics in many of the above areas are peculiar to computer security and are not likely to be covered in even the best intentioned laboratory activities of standard courses. How many standard courses will examine firewalls, guards, and intrusion detection tools, or will all allow students to conduct penetration studies, build covert channel exploitation mechanisms, or construct modules in a security kernel? Security *does* address special topics and security-specific laboratory exercises are needed to elucidate them.

4. TAILORING TO OBJECTIVES

Depending upon the objectives of the particular institution, laboratory exercises will emphasize one or more of these categories. A program designed to educate future network security administrators might focus on exercises using selected operating systems and tools and might have few, if any exercises exploring the technical foundations of computer security. This is logical since a system security administrator is unlikely to be required to know and understand formal logics for authentication protocols, just as an automobile mechanic can do his job perfectly well without knowing about the physics of the internal combustion engine. On the other hand, a program intended for future system designers needs to inform students on topics that will equip them to recognize products that would only work if the vendor had solved the Halting Problem. If our students are to construct new protocols or operating systems, then their laboratory exercises will focus less on products and configuration, and more on theory and design.

A conclusion from the above observations is that there is not one package of laboratory exercises that will work for all computer security education programs. Yet, we must be careful. Teaching students the details of a particular product or how they can fill a particular corporate security role, so that they can meet some set of corporate requirements or pass a standardized test may border on training rather than education. Even a college or university program that is acting as a feed to a large enterprise where specific security mechanisms and tools are predefined may not serve its graduates well in the long term if they know *how* but not *why*. My bias is to emphasize principles that graduates will carry with them throughout their careers. For example, in programming courses, students code in a particular language, yet they also learn concepts that carry over to other languages. We use specific tools to teach general concepts.

Is there an area of computer security education in which laboratory exercises would be inapplicable? Consider an extreme example. Suppose the program at the University of Betelgeuse is educating crypto-mathematicians. Do these students need to be forced into the laboratory as part of their education? The answer may not be clear-cut. If an individual is going to be a pure mathematician, perhaps exploring new areas of mathematics or showing how existing mathematics can be brought to bear on the problem of cryptography, then laboratory work as part of her education may be superfluous. On the other hand, if the individual is designing cryptographic algorithms that must be realized in either hardware or software implementations, then there will be performance and memory issues to consider. Without an understanding of how the underlying hardware and/or software operates, an impractical algorithm might emerge. Although some might argue that this understanding might be derived only from textbooks, one’s appreciation of low-level system architecture is likely enhanced with practical experience.

5. SELECTIVE LABORATORY WORK

The educational process will be a mix of theory and practice, lecture and lab, so a class might consist almost entirely of laboratory exercises or have very few. Certain concepts lend themselves to laboratory exercises, while others are best taught at the blackboard. That balance and type of laboratory work will result as the teacher determines the educational objectives of the class. If one is interested in teaching students how to administer either security protection mechanisms or security services such as firewalls, intrusion detection systems, or public key infrastructure services, then exercises on these topics are needed.

The amount of laboratory work will also be a function of the objectives. If graduates are to be at least minimally competent system administrators, then practice is essential. Although some people can learn by reading, for many, hands-on experience is needed; topics often do not become “real” until encountered in the real world. Borrowing again from programming, we know the importance of practice. Good programmers have practiced by writing many *different* types of programs.

Whatever course is taken, the development of laboratory exercises can be time consuming as products and standards are rapidly changing. The use of existing products and tools can help instructors build useful laboratory programs, but they will not eliminate a substantial time investment by the instructor.

In an introductory course students may not have learned enough computer science to be prepared to engage in extensive system development projects. Instead a series of short exercises may be most appropriate. In the Naval Postgraduate School program, we have a set of laboratory exercises, each of which is used to illustrate a particular concept being taught in class [3]. Each exercise includes questions that students must answer and our exams usually include a few questions related to the exercises. The exercises cover topics such as passwords, mandatory access controls, discretionary access controls, secrecy and integrity policies and their enforcement, viruses, steganography, cryptography and protocols, evaluation criteria, etc. We have found that for many students, the concrete exercises significantly clarify the concepts we are attempting to convey.

In advanced courses, student maturity permits more extensive laboratory projects. Classes on the secure management and administration of systems can include tests of various techniques and products. Since such tools are often those of the adversary as well as of system defenders, they provide students with two views of the system. Experimentation can also illustrate the challenges associated with configuration and management of various security mechanisms. Students can also use risk assessment tools and products intended to support system evaluation and accreditation.

In network security courses, the objective is to teach students how to combine heterogeneous components in to a coherent, distributed secure network. A project to describe the overall architecture and then "build" a secure network teaches students principles while familiarizing them with current tools and products. A miniature "world" with a public key infrastructure supporting a variety of protocols to accomplish the "real" work of applications will help students understand important concepts as well as introduce them to unsolved problems.

In courses examining the construction of security enforcement mechanism using software and hardware components, there are challenges for the instructor. Obviously the class cannot build a complete security kernel from scratch. One approach to laboratory exercises is to conduct extensive security testing and analysis of selected components of an existing operating system. Students can also study how secure systems are specified and constructed to avoid explicit and implicit interfaces that could be exploited by adversaries. The can build selected modules or can test various covert channel hypotheses.

Even course work on security policy models lends itself to the laboratory. Here the objective is not be to produce the developers of the next model of information flow, rather students should understand several existing models and have an appreciation of how models are used in the context of computer security. The fact that a formal model of security policy can be developed and proved gives many students a feeling of closure not possible in systems based on heuristic techniques. Exercises using a mechanical theorem prover allow students to understand how a set of high-level security requirements can be translated into a logical formulation and then proved to be consistent with its axioms over system transitions. In a progressive set of exercises, students may start by proving some rudimentary theorems from set theory and progress to develop a simple mandatory access control model and the prove its basic security theorem.

Demonstrations will still have a place in our curricula. Often students need to see how something is done before they can do it themselves. In other cases, we have yet to determine how to create an exercise that has the same pedagogical value as a good demonstration.

6. SUMMARY

Learning is not a passive activity. Study through lectures and reading requires the engaged student. Yet, the student who can learn a programming language merely by reading books and never coding is truly rare. As is the case with programming, listening and reading about computer security are not enough for most students: laboratory exercises must be a part of their studies. They help students understand and internalize key concepts. Exercises can force the student into a creative process using and building upon previous knowledge. New technologies are emerging that are likely to revolutionize the way we teach and conduct exercises. As this happens, we must ensure that students are engaged in active learning, not merely passively stepping through a few web-based demonstrations. Our creative use of new web and virtualization technologies will permit the creation of a new genre of exercises are will help us to teach complex and difficult concepts in computer security. These exercises will amplify the value of our lectures and better prepare our students for a complex and exciting future.

REFERENCES

1. Brinkley D.L., Schell RR. What is There to Worry About? An Introduction to the Computer Security Problem. In: Abrams, Jajodia, and Podell, ed. *Information Security: An Integrated Collection of Essays*, Los Alamitos, CA, IEEE Computer Society Press, 1995, pp. 11--39.

2. Irvine, C.E., Chin, S.-K., and Frinke, D., Integrating Security Into the Curriculum*, IEEE Computer*, V. 31, No. 12, pp. 25-30, December, 1998.
3. Irvine, C.E., Warren, D. F., and Clark, P. C. “The NPS CISR Graduate Program in INFOSEC: Six Years of Experience,” *In Proceedings of the 20th National Information Systems Security Conference*, pp. 22-30, Baltimore, MD, October 1997.
4. White, G. and Nordstrom, G., "Security Across the Curriculum: Using Computer Security to Teach Computer Science Principles," *Proceeding of the 19th National Information Systems Security Conference*, Baltimore, MD, pp. 483-488, October 1996.