Steve Zdancewic
Teaching Statement
September 4, 2007

CSE 331: Introduction to Networks and Computer Security

During the Fall semesters of 2002, 2003, and 2004 I have taught CSE 331, a junior- and senior-level undergraduate class that introduces the basics of networks and computer security. The course covers a series of topics including Ethernet, TCP/IP, denial-of-service attacks, cryptographic protocols for authentication and digital signatures, buffer overflows, web security, and malicious code. Enrollment in the course is typically 30 to 45 students, and the course work includes four fairly substantial programming projects.

One of the projects involves carrying out and then fixing a buffer-over attack against a simple C program. This project gives the students hands-on experience with techniques of malicious code and requires deep understanding of compilers and assembly language programming. A second project requires the students to implement an intrusion-detection system similar to a firewall. This project requires an good understanding of the TCP/IP network protocols as well as virus and worm signatures. In the third project, students implement a simplified version of the DES encryption algorithm and use it to crack some provided ciphertext. This project gives students a taste for what cryptographic algorithms are, and demonstrates why they should not rely on ad-hoc measures for creating secure systems. The final programming project is a simulated bank server and ATM system for which the students develop appropriate authentication and access control mechanisms in Java. This project emphasizes applications of cryptography and protocol design, and shows the students how to implement secure distributed systems.

Somewhat unusually for a computer science course, a significant portion of my midterm and final exams consist of essay questions in which the students are asked to critically assess the security measures of some real-world system (usually one taken from the research literature or current news articles). In addition to the usual problem-solving and knowledge synthesis questions that test basic understanding of the course concepts, these essays allow me to ascertain how well students are able to apply the principles of secure system design.

CIS/TCOM 551: Computer and Network Security

During the Spring semesters of 2005, 2006, and 2007 I have taught CIS/TCOM 551. In 2005 this course was co-taught with Matt Blaze. CIS/TCOM 551 is a graduate level version of CSE 331 that de-emphasizes the computer network parts of the syllabus and goes deeper into some aspects of malicious code, notably virus and worm propagation models and containment techniques. This course attracts approximately 30 masters level students and a few Ph.D. students each year.

CIS 670 and CIS 700: Graduate seminar courses

In the Spring of 2003, I taught the graduate seminar course CIS 670: Advanced Topics in Programming Languages–Safety and Security. The course covered topics of advanced language design, type systems, and program analyses as they apply to safety and security of software. I organized the course around a sequence of current and classic research papers and prepared lectures on each topic so that we could cover the material in depth. Students in the course worked in groups on research projects, which I supervised via additional meetings outside of class. The projects were intended to give students experience with writing research papers, and two of the projects led to workshop publications.

In the Spring of 2004, I co-taught a different graduate seminar course with Benjamin Pierce. This class, CIS 700: π-Calculus and the Foundations of Concurrent Systems, introduced Milner's π-calculus as a tool for studying key features of concurrent systems, including synchronization and message passing.

In the Fall of 2005, I taught a CIS 700 seminar on Software and Compiler Verification. This literature survey course was intended to provide a strong foundation in the historic and current approaches to verifying compilation, focusing on the programming language and compiler aspects of the problem, but also touching on related subjects such as theorem proving and software model checking.

Ph.D. Student-taught mini courses

Over the past few years I, along with C.J. Taylor, Fernando Pereira, and other Penn CIS faculty have developed a series of Ph.D. student-taught "mini courses". The mini courses are half credit classes that meet either for half a semester or just once per week for the full semester. They are intended to allow deeper coverage of programming languages and other topics that are useful in many contexts but that don't fit naturally in existing courses. To date, the mini courses topics have included C and C++ programming, Unix skills, Python, and C#.

My role in the mini courses has been to coordinate the Ph.D. students who do the actual teaching. This involves finding appropriately qualified students who are interested in gaining substantial teaching experience and then helping them to design the course structure and content. I also provide guidance about grading, course management, and interacting with students. The Ph.D. students are largely in charge of creating and presenting the lectures, generating homework assignments and quizzes, and managing graders for class projects.

My experience with these mini courses suggests that they are a real win–win situation: the Ph.D. students get first-hand practical experience with teaching, which is good for their resumes and helps them sort out their priorities with respect to pursuing careers in academia; the undergraduates get enthusiastic instructors who offer a variety of desirable courses. The feedback from undergraduates about these mini courses has been extremely positive and the Ph.D. students who have participated as instructors also report that it is a great experience.

The Oregon Summer School

For the past several years, the University of Oregon has hosted an advanced summer school centered around the topics of programming languages, formal methods, and security. The curriculum is designed to be an intense, two-week long course of study for Ph.D. students and researchers taught by experts in the various subject areas. In 2003 I was invited to participate as a lecturer in the series, and I gave three talks on information-flow security in programming languages. Subsequently, I was invited to be a co-organizer with Zena Ariola, of the University of Oregon, and David Walker, of Princeton University. In 2004, I participated as both organizer and lecturer, and I presented three talks on the Java and C# security models. In 2005, I participated as an organizer; the school focused on reliable and fault tolerant computing. Since that time, I have been on a steering committee for the summer school—it is currently petitioning to obtain permanent support from the ACM SIGPLAN organization.

These summer schools have been extremely successful in attracting bright students from around the world. The enrollment has climbed to approximately 40 students, and their feedback about the school has been extremely positive (90% of the students surveyed felt that the material taught at the school would directly impact their research). In addition to its educational aspects, the school has also facilitated contact among top researchers in the security and languages community with the next generation of researchers, providing opportunities for collaboration.