Teaching Statement

Jedidiah R. Crandall

5 February 2007

I see the material in many areas of computer science, particularly those that I would be interested in teaching, such as operating systems, security, and architecture, as having several levels of detail. At the high level students should understand general trends and be able to recognize themes in the material, while the low level entails a lot of detail that is necessary, not just for technical competence, but also so that the students have a strong sense of the general trends and themes. For example, in the Linux virtual memory management system there are many important themes: that various tasks are performed as late as possible, that page table entries are essentially capabilities, that increasing complexity can either help or hurt performance but always hurts security, and so on. To truly understand these themes the students should be familiar with all of the different cases a page fault handler must consider, all of the steps in the process of swapping a page in and out, proposed improvements for media applications, and many other details.

I have found that security can be a means (but certainly not the only means) of encouraging the students to study all of these things at length. How much more likely is a student to pay close attention through several lectures on virtual memory management and read a couple of chapters if you promise that at the end you will show them how the material can be applied for injecting malware into the kernel space of a system without loading it as a driver or using other such direct mechanisms? My personal experience giving guest lectures in several classes (both undergraduate and graduate operating systems, and program analysis) has been that you can indeed have students on the edge of their seats while talking about something as otherwise mundane as memory mapping a file. It is my belief, and something my dad who was an advertising, history, business management, and economics teacher for twenty years has often told me, that the only way to teach a student to have original ideas of their own is to give them a strong grasp of the related material since great ideas are mostly founded on the ideas of others. Copernicus discovered that the earth revolved around the sun only after rigorously comparing the Ptolemaic system with observed data and "rereading the works of all the philosophers which [he] could obtain." (Copernicus, *On the Revolutions of the Heavenly Bodies*, 1543)

In addition to teaching in the classroom, I would apply this philosophy to graduate advising, since having their own original ideas is so central to a graduate student's development. I have been very fortunate to have Professor Chong as an advisor, who always put my intellectual pursuits before his own interests even when my research began to stray from his main career focus, computer architecture. I have also had the pleasure of working closely with Professors Zhendong Su and S. Felix Wu. Professor Su works tirelessly to foster in his students an ability to come up with their own research ideas. Professor Wu is willing to discuss at length any ideas a student has no matter how outlandish they may seem. I hope to have the opportunity to return these endowments to my own students.

Teaching Interests There are a variety of classes I would enjoy teaching at either the undergraduate or graduate level, among these are:

Operating Systems: I have seen operating systems classes taught two ways. One is to cover all of the history and theory including processes and threads, different scheduling algorithms, synchronization, deadlocks, memory management, and filesystems. The other is to explain the implementation of an open source OS and then give students projects where they must "get their hands dirty" and implement their own kernel code that must integrate with the existing systems. I like both approaches and feel that both are important, so I would aim to strike a balance if I were given the opportunity design such a course.

Computer Security: Computer Security is a field that draws from a large and diverse set of different core competencies, meaning that there are many different forms a class on computer security might take. I very much enjoyed taking a graduate-level computer security course from Matt Bishop out of his book, "Computer Security: Art and Science." No matter how practical and down-to-earth my research became I always found myself referring back to the general principles of that book. I would try to design a security course around those principles but still reserve time to take students "down in the trenches" to do something real such as analyze evasive malware in Windows or subvert the content filtering of a Cisco router.

Hardware and Software for Security: Various hardware and software techniques have been proposed to make systems more secure. Understanding this long history as well as recent work is critical if the community is ever going to truly address this problem, which is why I would propose a class that is a deep literature survey covering vulnerabilities, subversion of system security mechanisms, memory protection, and perhaps also information flow security and malware. The course would survey papers over the past few decades and include recent work in the programming languages and architecture communities. I have several themes I would try to show the students, and then I would hope that they would surprise me by discovering their own themes in the literature.