Teaching Statement of Kirill Levchenko

1 Introduction

I teach both undergraduate and graduate classes related to computer security at the Computer Science and Engineering Department at UC San Diego, including the Department’s core graduate and undergraduate computer security courses. The main challenge is developing a course that covers the particular set of security topics an instructor considers important. While computer security is not a new discipline, with seminal works dating back to the 60’s and 70’s, it has undergone major changes in the last few decades as the nature of security threats we face and how we approach them has evolved. Traditional security focuses heavily on how to build absolutely secure systems from first principles, and this approach works well for small, well-defined elements of a bigger system. In their future, however, students will have to deal with very real and imperfect systems, attacked by an adversary who does not play by our rules.

A modern computer security course should be both relevant and practical, giving students direct experience working with, and implementing, the concepts covered in class. Unfortunately, there is no easy, ready-to-use off-the-shelf curriculum or textbook that satisfies these requirements. As an instructor I am responsible for deciding which topics to cover and how to cover them, a process that is both challenging and rewarding. In the remainder of this teaching statement, I describe my approach to teaching both undergraduate and graduate computer security, as well as graduate student advising.

2 Undergraduate

This fall is my second time teaching the Introduction to Computer Security (CSE 127) course at UC San Diego. (I previously taught CSE 127 in Fall 2014). A 10-week quarter necessarily means that some topics will have to be covered only briefly, if at all. My syllabus this quarter allocates three lectures and a programming assignment to each of the following: core concepts, Web security, control flow hijacking, applications of cryptography, network security, and miscellaneous (e.g. Bitcoin). Each of these topics can be a course in their own right, so the challenge for me as an instructor is to meaningfully condense each of these areas into three lectures.

Learning goals. My goals for the course are for the students to be able to reason about the security of a system, whether it be a Web service, network protocol, or executable. After taking the class, students should be able to identify the security risks in each of these domains and to be able to reason about potential defenses. For example, after covering control flow hijacking, existing defenses (e.g. ASLR and DEP), and attack techniques to circumvent those defenses (e.g. ROP), I expect students to be able to reason about the feasibility of potential control flow hijacking attacks and effectiveness of potential defenses.

My assessments (midterm and final) reflect this view. In the case of control flow hijacking, I propose a hypothetical defense and ask students to evaluate its effectiveness. On my last midterm, I asked students to consider a defense that consists of XOR’ing the stack pointer with the return address stored on the stack. I ask students whether this defense is effective at preventing stack buffer overflow attacks, and how defenses such as ASLR or DEP might affect an attacker’s probability of success. The rationale for this approach is simple: most students will enter the workforce as software engineers, where they will be confronted with security problems and will need to design and evaluate potential defenses, and I believe teaching them to think critically about attacks and defenses will prepare them for that.

Teaching methods. In addition to deciding which topics to cover in a class, another challenge is deciding how to teach them. One of the major decisions a computer security instructor must make is how much emphasis to place on attacks in lectures and programming assignments. For example, it is common for many undergraduate computer security classes to teach control flow hijacking by having students implement attacks against a set of target binaries. An alternative approach is to teach defenses only, without having students practice attacks. A defense-focused control flow hijacking assignment, for example, might ask students to find and fix potential control flow hijacking vulnerabilities in a sample program.
My own inclination, when I first taught the class, was to emphasize defenses over attacks on programming assignments. After all, most students will be on the defending, rather than attacking, side when they graduate. However, over time, my thinking on the matter has evolved to include more attack assignments. Attacks teach adversarial thinking, an ability to identify security weaknesses in a system by thinking as an attacker would.

There is another, more pragmatic, reason to teach attacks. In my experience, students are much more engaged in attack assignments. The most popular programming assignment in my undergraduate class is the control flow hijacking assignment. The assignment, developed by Hovav Shacham, asks students to exploit five vulnerable executables to gain root privileges on a VM we provide. Though many students spend upward of 20 hours on this assignment, most get a lot of satisfaction from being able to complete it (Figure 1).

The Web security project I developed for the class aims to balance attacks and defenses. The first assignment in the course is to develop a simple, Twitter-like Web application called Chattr. Chattr allows users to sign up and post publicly-visible messages to their page. By doing this, I make sure that students have the necessary Web development background to understand the Web security part of the class to be covered later in the course, giving them a chance to catch up independently if they need to. After covering password storage, cross-site request forgery, cross-site scripting, I gave students a two-part assignment. For the first part, students attack an instance of Chattr we have running. (This part of the assignment is similar to Stanford’s Zoobar assignment.) The the second part, I have students fix vulnerabilities in their own implementation of Chattr. (To aid them in this, the TAs provide a report from our automatic Chattr vulnerability tester.) This way, the students learn both how attacks really work, as well as how to make sure their own site is not vulnerable.

Teaching evaluation. I have received positive reviews of my teaching. Of the 87 students who completed my undergraduate computer security course in the Fall of 2014, over 90% said they would recommend the instructor.

3 Graduate

I have also taught the graduate computer security course at UC San Diego (CSE 227, Winter 2016). The goal for graduate computer security is to expose students to research in computer security, through paper discussions and a term project. For each lecture, students read two research papers on a related topic. During the lecture, I lead a directed discussion of the paper, aiming to cover the key ideas of the work. My goals are two-fold: first, to make sure students understand the paper, and, second, to learn to ask research questions.

One of the challenges with this class format is making sure students read the paper before coming to
class. There are several ways to encourage this. In my class, I do a short 5-minute, 10-question true/false quiz about the reading at the start of the class. This ensures students read the papers, and, as an additional bonus, arrive to class on time.

The term project is an opportunity for students to demonstrate their own ability to do research. Students work in groups on a topic they choose (often with some help or suggestions). The challenge is to identify topics that are, at once, tractable in a 10-week course, an actual research problem, and interesting to the students. I and my fellow instructors keep a list of potential projects for students to choose.

I also teach a security seminar every quarter. “Security Lunch” (CSE 229C), as the seminar is called, which takes place during the lunch hour, is an opportunity for the students and faculty to discuss the latest developments in computer security, alternating weekly between discussions of current research papers and current events. Devoting half of a seminar to news may seem a bit unusual, but it reflects the reality of modern computer security. Many news events have direct bearing on our discipline, from precedent-setting court decisions, to nation-state election interference, data breaches, and vulnerability disclosures. The news segment came about quite organically, as we found ourselves discussing current event items at the start of the seminar before discussing a research paper. Discussions and debates about current events would sometimes last well past the half-way point in the seminar, leading to the alternating schedule we have today. By devoting an hour-long seminar to news, students have an opportunity to research and present the technical background necessary to properly understand the news and its implications.

Finally, with my colleagues Stefan Savage and Hovav Shacham, I taught a special topics course, “Stuxnet to Snowden” (CSE 290, Spring 2015). Disclosures by whistleblower Edward Snowden have produced a lot of information about intelligence agencies’ cyber espionage capabilities, and this course was devoted to understanding these capabilities and their social and technical implications. Each seminar a group of students presented their findings on a specific topic or program disclosed in recent leaks. The goals of the course was to help students (and instructors!) place the revelations in a proper technical context and understand the full implications of the revelations. To understand the implications of the infamous “SSL added and removed here!” drawing, for example, it is necessary understand how companies like Google and Facebook store and process data, why data might be sent unencrypted internally, and the implications of this for users. Students maintained a wiki documenting their findings, which became the final product of the class.

4 Advising

I am currently advising 4 Ph.D. students, one of whom advised jointly with Stefan Savage. I have graduated one Ph.D. student, Danny Yuxing Huang, advised jointly with Alex C. Snoeren. Danny is now a post-doc at Princeton University. I have supervised 10 M.S. projects; my M.S. project students went on to employment at Microsoft, Google, Amazon, Broadcom, and Dropbox. Currently, I am advising one M.S. thesis student and one undergraduate student.

My advising philosophy is to help students become independent researchers. I hold weekly, hour-long meetings with each student to discuss their research progress. I also regularly see students outside of the weekly advising, as my door is usually open to impromptu meetings. Weekly advisor meetings are not a milestone that students must prepare for (and relax after), but part of a continuous process where an advisor guides the student toward a research goal.

5 Conclusion

Teaching and advising are some of the responsibilities and pleasures of an academic career. I direct the same energy toward my teaching as I do toward my research, because it is one of the most important ways in which we shape the next generation of scholars. In addition to computer security, I would also be interested in teaching courses in computer networking.