Teaching Statement

Dave Levin

I am committed to teaching, establishing inter-disciplinary collaborations, and supervising student research. I regularly seek out opportunities to refine my skills as a teacher and advisor. In my current position at the University of Maryland, I have taught two semesters of a senior-level undergraduate course on Computer and Network Security, a semester of a junior-level course on the Organization of Programming Languages, and a graduate-level seminar on Securing and Monetizing the Internet. When I was an undergraduate, I taught three semesters of discussion sections in undergraduate programming courses, for which I won an undergraduate teaching assistant award. I have also advised 25 undergraduates' research, and unofficially co-advised 10 graduate students. My experiences as a teacher and an interdisciplinary researcher have prepared me to be able to teach undergraduate and graduate courses on security, networking, and systems, and to advise students on a wide range of topics. Below, I share some of my philosophies and lessons learned from my experiences with working with students.

Engaging students in the classroom

In an age with online courses and freely-available lectures on YouTube, one has to ask: what role does a teacher have in a classroom? I believe that the role of teachers is to instill true curiosity in their students, and that this is best achieved through *interaction*. Interaction in a classroom cannot be matched by a book or a video on YouTube. I have experimented with various teaching techniques to determine how I can most effectively engage students during a lecture.

One of the ways I engage students is in the way I structure lectures. I believe in designing a lecture around a concrete problem and including students in the process of solving a given problem during lecture. I begin by presenting the context of the problem we will be covering: the history that led to the problem, and the importance of solving it. I try to spend as little time *lecturing* the students as possible, and instead focus on concisely presenting the *problem*. I then call on and encourage students to share their ideas on how to solve the problem, while evaluating and refining their ideas throughout the lecture. For instance, rather than simply lecture on how Tor works, I present the problem it tries to solve (anonymous communication), solicit and refine design ideas, so that by the end of the lesson, the students have effectively re-created Tor's design. Classes of students continually surprise me with their ability to solve a concretely defined problem when given the opportunity to do so. I believe these are the kinds of lectures that students remember, and that motivate them to pursue further work outside of class.

There is a trend of increasing class sizes in Computer Science; UMD's undergraduate enrollment in Computer Science has *tripled* over the past six years to 2,450 students. How can we continue to interact with our students in lecture halls of over 100 students? We can no longer expect to interact with every student in every lecture, but I have found that interacting with even just a subset of students can result in greater involvement from the entire class. I faced this question during one of the first courses I taught (a required junior-level course, with nearly 100 students in my section): I could barely see the students in the back row, let alone interact directly with them. During one of my lectures, I tacitly decided that the first four rows would be the "interactive" portion of the class, and began calling on students seated there. Throughout the course of the semester, more students started moving forward, more laptops closed, and more hands went up. Moreover, observing this form of dialogue taking place between the students and me helped all students (even those seated farther back) to better understand the material.

Another of my techniques to engage student interaction is to write on the board and to use slides as sparingly as possible during lectures. I find that writing on the board gives students time to process (and hopefully better grasp) the topics we are covering, and it allows me to adapt *how* I present the information based on the feedback I am getting during a class. Slides are fine tools for disseminating information: I restructured UMD's undergraduate course on Computer and Network Security, and, the slides and notes I created served the role as the class's "textbook." These slides have made it possible for my lecture materials to help shape others' offerings of the security course, as well as a Coursera course taught by Mike Hicks. Though useful in these regards, I find that slides tend to restrict the flow of a lecture. The lectures I give without slides are more interactive and, frankly, more fun. Finally, writing on the board forces me to walk into each lecture prepared anew, and allows me to easily adapt my lectures over time.

Engaging students beyond the classroom

Project assignments extend a lesson beyond the classroom and give students hands-on experience with what is covered in a course. I believe in a combination of programming assignments and more open-ended design problems. Program-

ming assignments give students the experience and familiarity with tools necessary to turn a high-level idea into a functioning system. Design-based assignments give students the experience of formulating their own concise problem statement. In my Computer and Network Security course, my early projects are mostly geared towards learning a specific set of techniques (e.g., buffer overflow and SQL injection attacks and defenses), though each one has subproblems for which there is not one specific answer. This prepares the students for the later, more open-ended projects, but it has also proven extremely effective at bringing out the creativity and engagement of the classes' top students.

I am also dedicated to creating new educational opportunities that are fun for students and provide empirical evidence for improved teaching. With colleagues from the University of Maryland, I have helped design and run the **Build-it/Break-it/Fix-it** secure programming competition (http://builditbreakit.org). This competition was motivated by a seemingly simple question: what concrete methods and tools help student programmers develop secure code? There are many competitions geared towards *attacking* (e.g., capture-the-flag), but, we wondered, how would students approach programming if they knew their code was going to placed under the scrutiny of attackers? BiBiFi consists of three phases: first, build-it teams implement a piece of relatively complicated code according to a specification we assign; then, we release all of their code to break-it teams, who attempt to find vulnerabilities; finally, the build-it teams get an opportunity to fix their bugs. We have held three full competitions, including hundreds of participants. To students, this is a fun opportunity to compete for prizes; to us, it is a way to gather data on what methodologies, languages, and tools work for writing secure code. Our goal is to use these findings to refine how we teach security.

The role of an educator has been, and always will be, to produce new generations of competent practitioners, innovators, and leaders. With the world's growing dependence upon computers, I believe that this task is all the more important within the field of computer science; our graduates develop the systems that dictate whether users' personal information is secure, build the networks upon which financial markets rely, and craft policies that determine the future demands and capabilities of the Internet. My goal as an educator is for my students to have a positive influence in whatever areas they pursue after graduation, not only by being able to build and adapt to new technologies, but also by having the confidence to identify poor policy decisions and the creativity to propose alternatives. My approach to designing lectures around collaboratively solving concrete problems provides the arena in which to instill these qualities in students. Additionally, in my security course, I start my first lecture and end my last lecture by telling my students that, after this class, they are responsible: their future users will depend on them to apply what they have learned.

Research experiences for undergraduates

I am dedicated and passionate about creating opportunities for undergraduate research, and advising undergraduates. In addition to personally advising 25 undergraduate students' research, I have recently been appointed co-chair of UMD's Computer Science Undergraduate Honors program. I will use my position as a tenure-track faculty member to cultivate research opportunities for undergraduate students.

Indeed, research begins in the classroom. I structure my courses around broad topics (e.g., my security course comprises software security, cryptography, and network security), and at the end of each of these, I always close with descriptions of ongoing projects and open problems as they relate to the topic I covered. This has resulted in several groups of students choosing the topics I present as their course projects, and involving themselves further with the lab once the semester has ended. Moreover, it shows concretely to undergraduates what constitutes computer science research: the kinds of problems we researchers investigate and the current state of what is and is not known to be possible. Making available the opportunities for continued research will continue to be one of my fundamental goals as a professor.

Advising

I have advised 25 undergraduate students, and worked closely with them on a wide range of topics—including security, network measurement, wireless networks, and human-computer interaction. This has resulted in 13 publications, and six undergraduates (so far) continuing onto graduate schools (including MIT, Princeton, and the University of Pennsylvania). I have also unofficially co-advised 10 graduate students. From these experiences, I have learned that each student is best advised in a personalized way; there are not many rules that apply to *all* students.

However, there are several constants that I do maintain when advising students. First, I make it clear to students what is expected of them—as a collaborator and as a scientist—and what constitutes good work. Second, I give them the freedom and encouragement to choose the problems they work on. I tend to suggest multiple projects (and encourage them to develop their own) but it is ultimately up to them; after all, they will do better work on a project they are passionate about. Third, before turning them loose on whatever problem they wish to solve, I work closely with them

to define evaluation metrics against which to measure the solutions we come up with. Without the means to weigh ideas against one another, an open-ended problem can seem daunting and slow down a research endeavor. Finally, I give my students the latitude to fail, and the time and feedback to turn those failures into lessons for future success. For example, I believe in editing my students' writing rather than generating it for them; the cost of many editing passes is far outweighed by the reward of the student learning to communicate his or her research effectively. These four approaches will help me achieve my goal as a professor: to produce independent researchers who are ready to pursue their own research agendas when they graduate.

The opportunities I have had to refine my teaching and advising methods are ones that I will extend to my students, as well. In particular, I will encourage my students to collaborate heavily with others and to establish reading groups for topics they wish to master; this was a piece of advice I received from one of my advisors early in my graduate career, and I believe it led to some of my most rewarding work. I have found reading groups to be invaluable; if done right, they can help keep a group up-to-date, yield new research ideas, and give a hands-on feel to students of how to think and talk about research. I have participated, and played a large part in running, many reading groups. At the University of Maryland, I ran a systems reading group (Syschat), started a game theory reading group (GTRG), and participated in reading groups at the Max Planck Institute for Software Systems (MPI-SWS), Microsoft Research (MSR), and started a reading group at HP Labs to encourage collaboration among different research groups. The formats of these varied from institution to institution, and particularly between academic and industry research settings. One that I have found most successful with students is to begin a meeting with a lecture-like talk—to give students experience speaking in front of a large group—followed by a short round-table discussion—to dive deeper into the work presented while fostering new ideas. It is difficult to quantify the precise benefits of such a format, but in my experience, it instills the students with skills to confidently evaluate others' work (as well as their own) and give stellar presentations.

I consider teaching and advising to be a privilege and a pleasure. While I have developed what I believe to be a successful style, I actively refine my approach by soliciting student feedback, discussing pedagogy with other educators, and adopting others' techniques into my own. I recognize that increasing class sizes and more opportunities for online learning will require me to refine my techniques further, and I welcome this challenge.