Owen Arden
Teaching Statement

1 Teaching experience

The opportunity to teach and mentor students in computer science is very attractive. I have sought out such opportunities during my time at Cornell, but also prior to that, while I worked at the National Security Agency. From 2004 to 2006, I was co-instructor for Software Vulnerability Analysis, a course offered twice a year by the NSA's National Cryptologic School. This course was part of the curriculum for two NSA career-development programs, the Cryptomathematics Program and the Applied Mathematics Program. During the summers 2004, 2005, and 2006, I designed security-related projects and mentored students as a Problem Supporter for the NSA Director's Summer Program, a program for outstanding mathematics and computer science undergraduates.

At Cornell, I was a teaching assistant for three courses. First, I was the head TA for the practicum portion of *Systems Programming and Operating Systems*, taught by Gün Sirer, in which I gave weekly lectures presenting the concepts necessary to build components of a simple operating system. I received high evaluation scores from the students and received the Teaching Assistant Award by the Department of Computer Science. I was also a teaching assistant for an introductory programming course, *Object-Oriented Design and Data Structures*, first with Graeme Bailey, and later for Andrew Myers for the honors version of the course.

Despite the differences in subject matter, I adopted a similar approach for both the introductory programming courses and the more advanced systems programming course. In my lectures, I had two goals. First, I tried to give the students a firm foundational understanding of a core concept such as inheritance or concurrency control. Second, I demonstrated concrete manifestations of the concepts in mechanisms like the Java class hierarchy or locks and semaphores, and discussed principles of using these mechanisms to design programs.

My experiences at the NSA have taught me that security is an excellent way to engage new students in computer science. Adversarial thinking pushes students to expand their understanding of difficult concepts and is effective for considering the implications of design decisions in many areas of computer science. Couching the consequences of design errors in terms of how a nefarious hacker could exploit them helped my students think more defensively during their design process. I also stressed to my students that an apparently “working” program is insufficient. Especially in concurrent programs, testing all possible behavior is very difficult. It is therefore important not only that a program work, but also that the implementation is understandable. Difficult-to-understand programs may hide serious vulnerabilities or other errors, or may cause confusion that result in erroneous modifications.

2 Teaching interests

Broadly, I am qualified and interested in teaching courses in security, programming languages, and systems. More specifically, I would enjoy teaching security courses that focus on topics such as systems security, formal methods, and language-based security. In programming languages, I would enjoy teaching introductory programming courses as well as more advanced courses on compiler design and implementation, program analysis, and programming language theory. In systems, I would enjoy teaching courses that cover systems programming, operating systems, and distributed systems.
I would also like to teach a seminar on language-based security that explores the tensions between security, functionality, and performance. This is a particularly interesting time to explore these topics—language-based security has been an active (and growing) area of research over the past few years. The security crisis in industry is very real, and addressing it involves challenging research problems. Thus, I think students would benefit greatly from staying abreast of the most recent results.

3 Mentoring

I have mentored two students at Cornell, Samarth Lad, an M.Eng. student now at Glassdoor, and Matt Loring, an undergraduate now at Google. Both students were working on projects related to the Polyglot extensible compiler framework and related languages. I advised Samarth for his M.Eng. final project, which significantly improved the performance of compilation of all Polyglot language extensions by implementing an in-memory file system to avoid disk writes during source-to-source translation. I advised Matt Loring for two projects. In one, Matt designed and implemented a secure wiki web application in the Fabric programming language. In the other, Matt extended the Jif compiler to support Java-style generic types. Matt’s “Generics in Jif” project was the winner of the PLDI 2014 Student Research Competition.

4 Programming Languages Discussion Group

I organized the Programming Languages Discussion Group (PLDG) at Cornell for two years. The seminar met weekly for presentations on current research given by Cornell students as well as PL researchers from both industry and academia. I started a new summer program where in lieu of discussing different research topics, the members of PLDG conducted an in-depth look at a particular topic of interest. These topics were typically selected to build critical background or skill-sets that students might find useful in their research, but might not be part of the Cornell PhD program’s core curriculum. For instance, in summer 2012, we covered Frank Pfenning’s Linear Logic course at the University of Pennsylvania using online lectures and exercises. In the summer 2013, we built skills in Coq by working through materials from Benjamin Pierce et al.’s and Adam Chlipala’s textbooks.