Shiqing Ma

Teaching Statement

One important reason for my pursuit of an academic career is that I enjoy mentoring and teaching students. I have been fortunate to have a number of inspiring teachers and advisers, and I truly look forward to inspiring my own students.

Experience

Guest Lectures: During my graduate study, I have acted as the guest lecturer or substitute lecturer for several courses including the graduate level CS529 Security Analytics (with Prof. Ninghui Li), CS510 Software Engineering (with Prof. Xiangyu Zhang), as well as the undergraduate level CS240 Programming in C (with Prof. Xiangyu Zhang). During teaching, I try to teach by examples. When I was giving lectures on adversarial samples for CS529, I showed the students adversarial images generated with 11 different methods from the same seed input, and highlighted their perturbations. This made it easy for me to explain these attacks and compare them, and students told me they loved the examples. I also showed them a few attack demos on auto-driving systems, and some students were interested and wanted to participate in our ongoing work. The other method I found effective is to build connections between abstract concepts and things that students are familiar with. For example, in C programming, many undergraduate students found the array concept is hard to understand. So I did a comparison with spreadsheets in Excel, and showed them how array operations are similar to Excel operations. It also saved a lot of time for me to explain one dimensional array and two dimensional arrays. Interestingly, many students got motivated to learn C programming as it was even more powerful than Excel.

Lab Design: I think that designing labs to give students hands-on experience is important in teaching. When my advisor Prof. Xiangyu Zhang was preparing the materials for a summer school in Australia, I also participated in designing a number of attack forensics labs. We packed everything including realistic Advanced Persistent Threat (APT) samples and attack forensics utilities to virtual machines so that students can skip the environment setup and focus on learning the new knowledge. Students can easily trigger the attacks or use the analysis utility tools with simple scripts we provided. This is to avoid the steep learning curve so that they will not get scared away. On the other hand, to ensure that students understand all key steps and techniques, we design a sequence of small labs, each of which contains a key technique. For example, in one lab, we ask the students to instrument a set of programs so that they can understand how program analysis can help improve the attack forensics. We carefully selected programs from different application domains (e.g., editors, video players and browsers) and different complexity (e.g., W3M browser and Firefox). This is to give the students a complete view of how it works on different types of programs, and also gradually improve their skills. After all labs are done, they will be chained together and students are asked to perform the whole attack and investigation process using the code they have written in the final lab. Many students felt proud of what they did and said they had learned a lot.

Student Supervision: Supported by my advisors, I initiated and supervised a group of four junior Ph.D. students to do machine learning research from the program analysis perspective. I also organize a weekly reading group with over ten regular members on security and software engineering. In addition, I am leading a reliable machine learning reading group with Professors Xiangyu Zhang, Suresh Jagannathan and Roopsha Samanta and over fifteen Ph.D. students. While supervising junior Ph.D. students, I believe the key is to understand them. So I always try to proactively talk to them, and we use various tools to communicate and collaborate such as Slack for project discussion and resources sharing, and Google Docs for sharing papers and reading notes. When started the supervision, I always tried to explain the low-level technical details of their current project, and then gradually talk about high level design and even research visions. For example, when I was working with Le Yu, a second year Ph.D., on a project which requires analyzing applications like Firefox whose code-base is extremely complex and he had no experience with, I spent many hours showing him the tools and methods I used by instrumenting the code base to solve a few simple problems (e.g., printing the current tab id and using \texttt{rr} to record and replay execution to understand the internals). We did this a few times, and now he is fully capable of doing the research by himself. After noticing that he spent less and less time on these low level technical problems, I start to bring up discussion topics like the framework design, its benefits and so on. This approach turns out to be very effective, and in the past two years, I have published many top-tier conference papers on NDSS, FSE, NeurIPS and so on together with these junior Ph.D. students.

Philosophy

I found three things are important for teaching: clarity, teacher-student interactions and examples. Thus my teaching approach will emphasize these three aspects.
Clarity: I believe one important way of improving clarity is to develop a sequence of specific and measurable goals. Students attend a class with their expectations, but many of them, such as “I want to learn C programming,” are too general and not clear. During weeks of study, students can get bored and lose track of the learning process. A sequence of goals sets milestones for the students so that they are aware of their learning process and know what to expect next. This helps improve the clarity of my course organization. Specific goals are concrete and executable tasks. For example, the goal “I want to learn C array.” is not specific. Instead, the goal “I will write an matrix computation program with C array.” is more concrete and executable. These goals will help students know where to start and how to achieve them, and my desire of teaching these materials can be clearly delivered to the students. Also, the goals shall be measurable such that students can easily know they have achieved it or not. The success of achieving a goal will help build the students’ confidence, and motivate the students. Failures will raise alarms for both the teacher and the student so that they can work on it together to improve it.

Teacher-student interactions: Building student engagement in classroom helps students understand the materials and focus. In class, I will include open discussion. I plan to use real world problems to start the discussion instead of purely conceptual questions. For example, when talking about the process scheduling policy goals in operating systems, I will ask “How will you design a policy for airplanes from United Airlines and American Airlines to take off from an airport with only one runway? What about 2 or more runways?”. These are problems that students have concrete understanding and can easily get into discussion, but denote the same essence as the textbook concepts. Meanwhile, I would encourage students to come up with such real world analogies such as the queuing schema in network systems versus the service scheduling in a bank. Such discussion will surely deepen their understanding. Besides introducing open discussion, I will also use the white board. During my writing on the board, students can take notes, digest what has been taught, think more, and ask questions. This will lead to more discussion and better understanding of the materials.

Examples: During my guest lectures, I found that live demonstrations and showing sample code that implements abstract concepts excite students. For example, when talking about the adversarial samples, I showed them a set of images and asked them to tell which ones are adversarial samples. Afterwards, I showed them the results from the target classifier, and then highlighted the perturbations. The students enjoyed the demonstration. They were so excited, and started to ask how this is done. I believe such concrete experience will help make the students curious about the techniques, and motivate the students to study them.

Teaching Plan

I believe I am qualified to teach a wide range of topics in both graduate level and undergraduate level including security courses (e.g., software security and machine learning security), software engineering courses (e.g., software engineering, program analysis, and software testing), and also introductory system and programming courses (e.g., operating systems, programming languages like C or Python, and compilers). In addition, I am ready to offer graduate level seminars on security and software engineering, which are directly related to my own research areas. I have a preference to teaching security courses and software engineering courses.

Security: For security courses, I would prepare materials for students to practise attacks, defenses and forensics. I already have a few virtual machines containing attack samples, defenses and forensics tools for both Windows and Linux. The labs designed with my advisor Prof. Xiangyu Zhang for the summer school can be reused and extended. Also, I plan to cover the security problems in deep learning systems including different attacks and defense/detection techniques, which represent one of my research interests.

Software Engineering: For software engineering courses, I plan to cover topics in traditional software engineering such as software testing, analysis and debugging. I will include both source code level analysis and binary level analysis using popular frameworks like LLVM, PIN, Valgrind etc., as well as emerging topics in data engineering such as deep neural network testing and verification.

Others: I am also interested in teaching classic computer science courses such as Operating Systems, Compilers, and Programming in C/Python. I plan to add some materials to the courses based on my research experience. For example, for Operating Systems, I plan to ask students to add the logging capability to the kernel for provenance tracking. For Compilers, I plan to design labs that require the students to write LLVM passes for simple instrumentation tasks. I also have an interest in organizing a new seminar course covering topics that are related to my own research, namely, security and software engineering problems in both traditional systems and emerging machine learning systems.