Teaching Statement

Penghui Li

Academic life offers a unique opportunity to interact with and influence future software engineers and computer scientists. I am enthusiastic about helping students develop not just technical proficiency, but the foundational judgment and mindset needed to lead and innovate. In an era where LLMs can generate exploit code and debug programs, students often ask me whether ChatGPT will replace security engineers. Rather than seeing this as a threat, I see an opportunity to prepare them for roles where AI enhances their capabilities but cannot replace their adversarial reasoning, systems thinking, and security judgment.

Teaching Experiences. During my Ph.D. studies at the Chinese University of Hong Kong (CUHK), I served as a teaching assistant for four courses, including linear algebra, web application development, database systems, and computer security. Together with instructors, I designed assignments and projects for about 200 students, graded submissions, and held regular office hours to help students clarify both conceptual and practical problems. When the COVID-19 pandemic forced the switch from in-person to virtual teaching, I worked closely with instructors to experiment with online tools and restructure assignments for remote learning. Beyond my TA responsibilities, I was invited to deliver guest lectures on agentic program analysis and web security in multiple courses, where I introduced the foundations of security vulnerabilities, demonstrated real-world exploits, and discussed the latest research. These experiences helped me develop a flexible teaching style adaptable to different topics, audiences, and formats.

Mentoring Experiences. I have mentored 11 undergraduate and graduate students at CUHK and Columbia, covering both short-term and long-term research projects. I meet with students weekly to discuss progress, brainstorm solutions, and set milestones. My mentoring emphasizes three principles: helping students identify research problems that align with their interests, analyzing the fundamental causes of security challenges, and encouraging independent problem-solving. This approach has yielded strong outcomes. For example, Zeyang Zhuang published a first-author paper on graph database testing at VLDB '24, Changhua Luo led a CCS '24 paper on Node.js exploit generation, and Yanting Chi completed a bachelor's thesis on symbolic execution and is now pursuing a Ph.D. at the University of Minnesota. Beyond publications, I take pride in mentoring students toward independent thinking, including skills such as debugging complex analysis tools, designing experiments, and presenting research clearly.

Teaching Philosophy. The rise of large language models (LLMs) is transforming the software engineering landscape and the skills that employers value most. While LLM tools can now assist with many routine programming tasks, they have also elevated the importance of deeper competencies such as system design, security reasoning, critical evaluation, and sound engineering judgment. As educators, we have an opportunity to prepare students not just to use these tools, but to excel in roles where human expertise remains essential.

My teaching philosophy centers on three principles that prepare students for this evolving landscape. First, *motivation through real-world relevance*. Students learn best when they understand why the material matters. In an undergraduate database systems course where I was a TA, students responded enthusiastically when lectures began with examples drawn from real-world applications such as e-commerce or social media platforms. These examples helped them connect abstract concepts to concrete impacts, motivating them to master technical details. In my future teaching, I will continue to design lectures around compelling examples that show how security principles apply to systems students interact with daily.

Second, *cultivating independent learning and critical thinking*. Computer science evolves rapidly, and programming languages, analysis tools, and security practices change every few years. Rather than simply transmitting knowledge, I aim to train students in problem-solving and inquiry. As a TA, when students approached me with debugging issues, I often refrained from giving direct answers. Instead, I provided guiding questions, suggested related concepts, and encouraged them to explore documentation or small-scale experiments. This method, though slower in the short term, helped students build confidence in their ability to find solutions. Several students told me afterward that this approach changed how they tackled both coursework and independent projects. In an era where AI tools can provide instant answers, this skill becomes even more critical, as students must learn to evaluate solutions critically, understand their limitations, and make informed decisions rather than accept outputs blindly.

Third, *developing skills that complement AI capabilities*. Today's graduates benefit most from competencies where human judgment remains essential. While LLMs can generate code, they still struggle with adversarial reasoning, security-critical decisions, and anticipating attacker behavior. I want to help students develop both offensive and defensive mindsets, cultivating the systems thinking and creative problem-solving that AI cannot replicate. For example, I plan to assign projects where students first use ChatGPT to implement a web authentication system, then conduct security audits to identify vulnerabilities the AI missed or introduced. This exercise teaches students not just to find bugs, but to recognize the systematic blind spots of AI-generated code, such as subtle timing vulnerabilities, inadequate input validation, and failure to handle edge cases that attackers exploit. Students will also compare their manual security reviews with AI-assisted analyses, learning when to trust AI outputs and when human oversight becomes critical. I emphasize that effective use of AI tools requires understanding their limitations—students must develop the judgment to know which tasks benefit from automation and which demand careful human reasoning. Beyond immediate coding tasks, I focus on architectural thinking and high-level design decisions, areas where human judgment and experience remain indispensable. By teaching students to critically evaluate and supervise AI outputs rather than accept them blindly, I prepare them for careers where their value comes from expertise that technology enhances but cannot replace.

Teaching Interests. I am prepared to teach courses in computer security and software engineering at both undergraduate and graduate levels. I am particularly interested in teaching Introduction to Computer Security, Web Security, Language-Based Security, and special topics courses such as Agentic Software Security. In these courses, I will apply my teaching philosophy by engaging students in hands-on projects that blend traditional security concepts with contemporary tools, such as implementing authentication systems, conducting penetration testing, or performing security audits. For example, in a graduate seminar on Agentic System Security, students would investigate emerging threats such as prompt injection attacks and tool-using AI vulnerabilities. Each course would include open-ended projects where students identify problems that matter to them, conduct original analysis, and present findings. This will prepare students not just to apply established principles, but to pioneer solutions to emerging challenges.