Teaching Statement

David Kohlbrenner

I’ve had the privilege to be the instructor and course planner for an undergraduate pilot course on systems and command-line interfaces (CLI) at UC San Diego and for a graduate-level secure hardware course at UC Berkeley. I also have experience from TAing a 200+ undergraduate security course, as well as serving as a teacher and mentor for 5 years to the UCSD undergraduate CTF (Capture the Flag — competitive hacking) team I founded in 2013. Teaching was one of the best parts of my PhD, and all of these experiences have been both educational and rewarding.

I have received exceptionally strong positive feedback for both my TA work and instruction. Anonymous reviews for my TAing of undergraduate security included comments such as “Extremely good in technical aspects. Takes time to explain the concepts even if the student is missing basic knowledge” and “David was extremely willing to repeat explanations and explain the necessary background to help students understand what was going on. He would stay after section until all questions were answered and never showed any impatience or frustration at the questions students asked.”

Introductory systems and CLI: In the summer of 2017, Joe DeBlasio (another PhD student) and I designed and developed a new undergraduate systems course we call “terminals for computer scientists” (T4CS) to teach basic terminal usage, bash scripting/tools, and general UNIX-like familiarity. The course was motivated by a problem we both observed in upper-level courses where some undergraduates were very uncomfortable with CLIs and terminal usage and had an incorrect mental model of lower-level systems. Consistently, these students would struggle with tools like ssh, scp, and basic file manipulation due to their lack of familiarity with how they functioned. T4CS is targeted at early-undergraduate CS and non-CS majors who will regularly interact with Linux/etc. (e.g. information science, bioinformatics) and who have no experience with command-line interfaces. Our hope was that it could help students with less background in tinkering with computers build confidence to continue into low-level systems courses.

T4CS is designed as “immersion learning for the terminal” with a focus on automation, rapid feedback cycles to students, introspection for instructors, and scalability. We took strong inspiration in our topics from the well-regarded “Computing for Computer Scientists” (C4CS) course at the University of Michigan. Some topics T4CS covers include: history of the terminal, basic shell interactions (ls, mv, cp, man, echo, etc.), basic vim, writing a shell script, pipelining, regexes, version control, the UNIX model, permissions, and productivity tools. To implement the “immersion learning” approach, the entire course was only accessible from a terminal via ssh; all in-class assignments, homeworks, and exams used an autograding system modeled on standard Linux tool interactions. All lectures were designed as follow-along with the lecture being performed live in a projected terminal. Grading was done automatically with instant feedback and hints for common mistakes. We implemented extensive data collection and replay capabilities for the instructors. These tools allowed us to immediately identify common mistakes or remotely assist students with homework. One notable benefit was that it let us identify difficulties students were having and proactively address the entire class without a student needing to ask a question.

Feedback on our pilot version of the class was uniformly positive. The T4CS infrastructure is available open-source on bitbucket with a full course of assignments and exams available on request to instructors. I am especially excited to collaborate with researchers in computer science education to use the data collection platform CSE80 offers to improve both T4CS and other classes. T4CS is fully transparent to, and regularly notifies, students about the data collection it performs and we recommend instructors discuss the data collection with students before starting. See the course syllabus at [https://cse80.github.io](https://cse80.github.io) and course infrastructure at [https://bitbucket.com/t4cs/t4cs](https://bitbucket.com/t4cs/t4cs)

Graduate secure hardware: As a postdoc at UC Berkeley, I planned and co-taught a graduate course on secure hardware. The course was a paper reading and large project graduate class targeted at early PhD students. A number of valuable projects resulted from this class including verification efforts for our Keystone project and experimental Spectre mitigations for Berkeley’s out-of-order RISC-V core.

CTF Advising As an undergraduate, I was introduced to security after I began participating in security competitions (CTFs) on the Carnegie Mellon team. It is unlikely I would have ended up as a security researcher and obtained my PhD without those experiences. I have found that for many students, CTF is a compelling way to start in computer security. It offers a unique chance to get hands-on with an
enormous variety of security topics in a team environment. During my time as a PhD student at UCSD, I founded and advised an undergraduate team. I believe that the team was a success: 20+ students joined the team at various times, many began security research, and a number eventually accepted security jobs in industry. Mentoring the students on the team offered me the chance to hone educational skills with a small and engaged group of students who could immediately begin applying the material. I generally provided technical information or background in short lectures on topics such as block cipher modes, return-oriented programming exploits, or others as they came up in challenges. I hope to continue supporting undergraduate CTF teams as a future faculty member.

**Teaching philosophy:** I strongly believe that one of the most important things a CS education can do is *de-mystify computers*, and T4CS tries to do just that. My experiences as an undergraduate and my time mentoring junior students have reinforced that there is no substitute for “actually doing it myself,” especially for security and systems projects. Consistently giving students the knowledge needed and opportunity to interact with low level system components allows them to build up accurate mental models of how computer systems function. These mental models will then serve them well as they encounter more complex topics in higher-level courses. Relatedly, the piece of feedback I am most proud of receiving was an anonymous comment on my TAing: “He has a great ability to provide useful insights without giving away the answers.”

I served from 2016 to 2018 on the UCSD Academic Integrity Review Board (AIRB) and hope to continue similar roles in the future. I have found that clearly establishing the bounds of acceptable behavior in coursework and enforcing those limits is critical to a fair classroom.

I look forward to teaching both my expertise area (computer security) at the undergraduate and graduate levels, as well as continuing to develop and teach early systems courses like T4CS. I have additional interest in teaching graduate seminars on the history of computer security and ethics as it applies to the field. I can also teach courses in operating systems, systems programming, C programming, introduction to computer architecture, or other similar courses. Examples of such undergraduate courses include UCSD’s CSE 30, 15, 12, 80, 120, 127, and 141.