Improve Cybersecurity Education by Bringing Secure Coding to CS1

New Approaches to Cybersecurity Education (NACE) Workshop, June 9 & 10, New Orleans, LA

Simson L. Garfinkel

The United States is utterly dependent on information technology, but only a fraction of the those working in computing specialize in cybersecurity. The reason is that the field of computing is tremendously broad. Just as there are now dozens of cybersecurity specializations, there are now dozens of computing specializations as well.

Consider the numbers from the 2016 Taulbee Survey, the annual survey by the Computing Research Association that tracks PhDs in computer science, computer engineering and information.[1] Of the 1888 students graduating in North America with a relevant PhD in 2016, just 106 (5.6%) found employment in "security/information assurance" — yet "security/information assurance" was the second largest employment category reported on the survey (only exceeded by Artificial intelligence). There are simply too many aspects of computing systems that require teaching and researching: security is critical, but so are the other specializations.

If our goal is to improve the state of cybersecurity using the lever of education, then we must consider ways of broadening cybersecurity education to include non-specialists. That is, we need a longer lever. This means incorporating security education throughout the entire computing curriculum, starting with the first computer science course that students take, affectionately called CS1 in the literature.

It has long been observed that many CS1 courses have programming examples that contain serious, exploitable security errors. In the days of "C" it was common for instructors to present programs with buffer overflow errors. These days, it is common to present programs that allow for brute-force password guessing, or SQL injection attacks, or just horrible usability that promotes unsecure use. We also have poor security practices in many educational computing environments—such as easy-to-guess passwords, open services, web services protected by hidden URL, and so on—in the interest of expediency.

[1] https://cra.org/crn/wp-content/uploads/sites/7/2017/05/2016-Taulbee-Survey.pdf

Programming examples with vulnerabilities and poor security practices in these introductory courses is poor pedagogy. We shouldn't be teaching the students with practices that we wouldn't want them to repeat on the job. We must scrub introductory courses of poor examples, and instead assure that these courses demonstrate good security practice. This will almost certainly require that security faculty partner with other faculty who teach the introductory courses.[2] [3]

As the need for programmers continues to expand, programmers who do not have the benefit of formal security instruction will be creating most of the code that powers our society. These programmers will use the tools of their trade. If introductory courses incorporate sophisticated security technology, it will be reflected in popular tools, there will be a multiplier effect. The result will be more code with fewer exploitable defects.

Other modern software engineering practices have been incorporated into introductory courses with great success, including test-driven development, continuous integration, and distributed source code control. These practices have been adopted because they make programmers more efficient and decrease software defects—and in the process, help to make software more secure.

Likewise, introductory programming courses should teach code annotations to support model checking, the use of static code checkers, and lightweight formal methods.[4] These techniques will be sold to students (and their teachers) as ways to make software more reliable and software development more efficient. As a side effect, their code will also be more secure.

April 26, 2018

[2] K. Nance. "Teach them when they aren't looking: Introducing security in CS1." IEEE Security and Privacy, 7(5):53–55, Sept. 2009.

[3] V. Pournaghshband, "Teaching the Security Mindset to CS 1 Students," SIGCSE'13, March 6-9, 2012, Denver, CO.

[4] K. Schaffer, J. Voas, "Whatever Happened to Formal Methods for Security," IEEE *Computer,* August 2016.