International Society of the Learning Sciences

A Dichotomy of Demonstrated vs. Perceived Knowledge in Computer Security

Michelle Jensen, University of Wisconsin - Madison, mejensen5@wisc.edu

Abstract: One approach to teaching computer security in undergraduate computer science (CS) programs is to enforce and distribute it across the curriculum. Topics accessible at even the earliest of levels (CS0, CS1, CS2) have been identified and tools studied to teach them. However, we do not know much about the process, understanding, or knowledge that students bring in when doing computer security without any coursework in computer security. We employ a series of think-aloud programming sessions and short interviews following the format of Build-it, Break-it, Fix-it (BIBIFI) competitions. In this paper, we take a closer look at one build-it session in which the actions and words of the participant conflict with their perception of being capable of doing computer security.

Introduction

A decade ago, the Joint Task Force on Computing Curricula (2013) added security and information assurance as a knowledge area to the curriculum guidelines document. In the years since we have tried to understand how to teach security to undergraduate students. Security has been positioned uniquely in programs in comparison to other core CS topics. Programs can generally fall into one of two categories: the traditional track or the security focused track. This can emphasize the idea that security is a technical or specialized set of knowledge despite it applying to a myriad of different situations in the computer sciences.

One dimension from prior work in computer security education and undergraduates, which we focus on, is distributing computer security across the curriculum (Blair et al., 2020; Siraj et al., 2015). One way to do this is by teaching students security-specific skills such as threat modeling to create a security mindset. We have identified different topics that are accessible to students who are in CS0, CS1, and CS2 classes (B. Taylor & Kaza, 2016). The current state of research into distributing across the curriculum (e.g. Locasto, 2009; Nance, 2009; Pournaghshband & Pournaghshband, 2021) generally evaluates using quantitative methods to determine the effectiveness and outcomes of interventions. This largely ignores one key part of the learning process; that students are not an empty slate. We know little about how students think and process computer security when given these sorts of tasks. In our study, we use a series of think-aloud coding sessions and interviews to understand students' perceptions and processes of doing security, regardless of if they have experience.

Theoretical framework

One area that we draw on to frame our research is the learning sciences theory of constructivism. This is important considering that there are relatively few examples of constructivist computer security education research in the literature. The other major part of our framework comes from Build-it, Break-it, Fix-it (BIBIFI) competitions. Andrew Ruef, James Parker, and Michael Hicks introduced BIBIFI competitions as another security-based competition in contrast to CTF (Capture the Flag) competitions. In BIBIFI competitions, teams across three phases create a (hopefully) secure application, attack other implementations, and fix vulnerabilities in their own. Research surrounding BIBIFI focuses on finding and categorizing the kinds of vulnerabilities that people make (Votipka, Fulton, Parker, et al., 2020). BIBIFI has also been implemented in a few classrooms; however, the research in these contexts says nothing about the learning the students do in the process of this activity (Fulton et al., 2022). In our work, we make a variation on the BIBIFI contest. Using the different phases of BIBIFI allows us to see participants' processes while they take on both attacking and defending roles.

Methodology

We employ a list of qualitative methods to examine students' thinking about computer security through a series of think-aloud work sessions with moderate participatory observations followed by short semi-structured interviews. We survey participants to get their background with computer security experience (although it is not required) and a list of completed CS classes. Using a simple Likert survey we also gauge their confidence in their programming, creating a bug-free program, and their capability in computer security. The first session mirrors the build-it phase. Participants are given a prompt to help implement a passcode save system for a simple text-based adventure game. They are given basic rules of the game and told that they should prevent the player from cheating. There are many different avenues that participants can take to work on this task, but it has been designed in such

a way that even those who have only completed foundational CS concepts can attempt to solve the problem. These were based largely on the topics identified by B. Talyor and Kaza (2016).

Results and discussion

We look in more detail at one build-it session which was selected based on the dichotomy of the participant's words and actions. While more intense, it is representative of a theme emerging in other sessions. The participant is a senior at the university who cited no prior security experience. Although confident in other areas of their programming, they "Strongly Disagreed" with the statement "I am capable of doing computer security." Comparing their response to words and actions observed during the coding session, it struck us as surprising. When asked to elaborate in the interview they stated they did not understand any of the technical details or skills such as the mathematics behind cryptographic methods or being the one who came up with an exploit. Despite this, the participant displayed a degree of security knowledge and skills in two different areas. The first area that they displayed security knowledge was in specific applied security topics (e.g. cryptography) and common attacks (e.g. buffer overflow). The second area they demonstrated was some semblance of a security mindset (e.g. defensive tactics, threat perception). We look to facilitate discussion on how to conceptualize the reason behind this observed dichotomy. Additionally, we also want to explore approaches to utilize this knowledge to better empower students in their current computer security knowledge.

References

Blair, J. R. S., Chewar, C. M., Raj, R. K., & Sobiesk, E. (2020). Infusing Principles and Practices for Secure Computing Throughout an Undergraduate Computer Science Curriculum. *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 82–88. https://doi.org/10.1145/3341525.3387426

Fulton, K. R., Votipka, D., Abrokwa, D., Mazurek, M. L., Hicks, M., & Parker, J. (2022). Understanding the How and the Why: Exploring Secure Development Practices through a Course Competition. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 1141–1155. https://doi.org/10.1145/3548606.3560569

Joint Task Force on Computing Curricula, A. for C. M. (ACM), & Society, I. C. (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery.

Locasto, M. E. (2009). Helping Students 0wn Their Own Code. *IEEE Security and Privacy*, *7*(3), 53–56. https://doi.org/10.1109/MSP.2009.66

Nance, K. (2009). Teach Them When They Aren't Looking: Introducing Security in CS1. *IEEE Security & Privacy*, *7*(5), 53–55. https://doi.org/10.1109/MSP.2009.139

Pournaghshband, V., & Pournaghshband, H. (2021). Teaching Security Notions in Entry-Level Programming Courses. *2021 IEEE International Conference on Engineering, Technology & Education (TALE)*, 997–1000. https://doi.org/10.1109/TALE52509.2021.9678545

Siraj, A., Taylor, B., Kaza, S., & Ghafoor, S. (2015). Integrating Security in the Computer Science Curriculum. *ACM Inroads*, *6*(2), 77–81. https://doi.org/10.1145/2766457

Taylor, B., & Kaza, S. (2016). Security Injections@Towson: Integrating Secure Coding into Introductory Computer Science Courses. *ACM Trans. Comput. Educ.*, *16*(4). https://doi.org/10.1145/2897441

Votipka, D., Fulton, K. R., Parker, J., Hou, M., Mazurek, M. L., & Hicks, M. (2020). *Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It*. 109–126. https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-understanding