Understanding security mistakes developers make: Qualitative analysis from Build It, Break It, Fix It

Submitted by Katie Dey on Thu, 05/09/2019 - 12:08pm. Contributors: Daniel Votipka Kelsey Fulton James Parker Matthew Hou Michelle L. Mazurek Michael Hicks

In spite of significant effort by the security community, the number of vulnerabilities reported in production systems has continued to increase. Automated vulnerability identification provides a promising path forward, minimizing the human effort and expertise needed. For example, recent developments in fuzzing to find memory errors or static and dynamic analyses to find improper uses of sanitizers have proven useful at finding large numbers of vulnerabilities quickly. However, the potential impact of these techniques remains an open question. That is, are current tools able to find most, or even a significant fraction, of the security bugs that are introduced by developers? If not, can existing techniques be adapted to account for missing bug types, or must entirely new techniques need to be developed?

To answer these questions, we first need to determine what mistakes developers commonly make that lead to vulnerabilities. Unfortunately, the data necessary to begin this line of inquiry is difficult to obtain. CVEs generally do not provide enough context about the vulnerability to identify the root cause and whether a current tool might find it [1]-[3]. Additionally, the conditions of development of the software (e.g., developer experience, organization's prioritization of security, etc.) likely have an effect which cannot easily be measured in an analysis of real-world code artifacts. To control for these effects, researchers have asked developers to solve simple security-relevant challenges in a lab setting [4], [5]. However, the challenges tend to be simplistic, and do not simulate real-world considerations (e.g., performance vs. featurefulness vs. security) well.

In this talk, we will present the results of a rigorous analysis of the code artifacts produced by four iterations of the Build It, Break It, Fix It (BIBIFI) competition [6]. BIBIFI asks developers to write relatively large programs with security-relevant features, and grades these programs on correctness, efficiency, and security, aiming to simulate real-world concerns. Usefully, the contest's design eliminates many variables that might confound results. We have many implementations of the same task, but teams were free to vary their design and implementation strategy (e.g., the programming language choice was up to them). This means that we can potentially see not just the presence of vulnerabilities, but their relative likelihood under similar circumstances. Over the four iterations of the competition, we have used three different task specifications, allowing us to study a wide range of possible issues.

We will discuss our analysis of 76 BIBIFI submissions and the 172 vulnerabilities they included. To develop an in-depth understanding of each vulnerability, we manually reverse engineered each project and utilized an iterative open-coding procedure to categorize each project and vulnerability. From this analysis, we found that underlying issues could be grouped into one of five categories: not attempting to provide a obviously or implicitly necessary security mechanism; attempting to provide security, but choosing a weak implementation; choosing a correct implementation, but misunderstanding how to correctly use a library; or making a mistake when implementing security (e.g., typo or forgetting a necessary algorithmic step). In the talk, we will discuss the prevalence of, factors potentially influencing the occurrence of, and recommendations for future work to prevent each issue class, and notably how automated tools and processes might assist in the process.

References:
Daniel Votipka is a PhD student at the University of Maryland studying computer security, with an emphasis on the human factors affecting security workers. His work focuses on understanding the processes and mental models of professionals who perform security-related tasks such as secure development, vulnerability discovery, network defense, and malware analysis to provide research-based recommendations for education, policy, and automation. Prior to beginning his PhD, he earned his MSISTM from the Information Network Institute at Carnegie Mellon University and served four years in the US Air Force as a Cyber Warfare Officer assigned to the National Security Agency.