Supporting Effective Strategies for Resolving Vulnerabilities Reported by Static Analysis Tools

Justin Smith
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: jssmit11@ncsu.edu

***Abstract*—Static analysis tools detect potentially costly security defects early in the software development process. However, these defects can be difficult for developers to accurately and efficiently resolve. The goal of this work is to understand the vulnerability resolution process so that we can build tools that support more effective strategies for resolving vulnerabilities. In this work, I study developers as they resolve security vulnerabilities to identify their information needs and current strategies. Next, I study existing tools to understand how they support developers' strategies. Finally, I plan to demonstrate how strategy-aware tools can help developers resolve security vulnerabilities more accurately and efficiently.**

I. Introduction

Static analysis tools, like Coverity [1] and Findbugs [2], enable developers to detect security vulnerabilities early in development. These tools locate and report on potential software security vulnerabilities, such as SQL injection and cross-site scripting, even before code executes. Detecting these defects early is important, because long-lingering defects may be more expensive to fix [3]. According to a recent survey by Christakis and colleagues, developers seem to recognize the importance of detecting security vulnerabilities with static analysis; among several types of code quality issues, developers rank security issues as the highest priority for static analysis tools to detect [4].

To actually make software more secure, however, static analysis tools must go beyond simply detecting vulnerabilities. These tools must be usable and enable developers to resolve the vulnerabilities they detect. As Chess and McGraw argue, "Good static analysis tools must be easy to use, even for non-security people. This means that their results must be understandable to normal developers who might not know much about security and that they educate their users about good programming practice" [5].

Unfortunately, evidence suggests existing tools are not easy for developers to use. Researchers cite several related reasons why these tools do not help developers resolve defects, for instance, the tools: produce "bad warning messages" [4] and "miscommunicate" with developers [6]. As a result, developers make mistakes and need help resolving security vulnerabilities due to the poor usability of security tools [7].

978-1-5386-4235-1/18/$31.00

Recently, Acar and colleagues introduced a research agenda for improving the usability of security tools, explaining that "Usable security for developers has been a critically under-investigated area" [8]. The goal of this thesis is to investigate and improve the usability of security-oriented static analysis tools so that we can ultimately enable developers to create more secure software.

II. Vulnerability Resolution Strategies

My thesis studies the usability of security-oriented static analysis tools through the lens of *vulnerability resolution strategies*. Building on Bhavani and John's definition of a strategy [9], we define a vulnerability resolution strategy as: a developers' method of task decomposition that is non-obligatory and directed toward the goal of resolving a security vulnerability. My thesis argues that tools can better help developers resolve vulnerabilities by presenting effective *vulnerability resolution strategies* alongside the defects they detect.

III. Usability of Static Analysis

Outside the domain of security, researchers have studied the human aspects of using static analysis tools to identify and resolve defects. Muske and Serebrenik survey 79 studies that describe approaches for handling static analysis alarms [10]. They organize existing approaches into seven categories, which include "Static-dynamic analysis combinations" and "Clustering." Sadowski and colleagues [11] report on the usability of their static analysis ecosystem at Google, Tricorder. Their experiences suggest that warnings should be easy to understand and fixes should be clear, which motivates the work in this thesis. Similarly, Ayewah and colleagues describe their experiences running static analysis on large code bases. They make suggestions for how tools should help developers triage the numerous warnings that initially might be reported [12]. In comparison, our work focuses on how developers resolve individual security vulnerabilities.

IV. Evaluation Plan

Phase 1 (Complete): What information do developers need while using static analysis tools to diagnose potential security vulnerabilities? To understand developers' information needs while using a security-oriented static analysis tool,

I conducted a think-aloud study with ten participants [13]. I observed participants as they assessed four potential security vulnerabilities using Find Security Bugs [14], a security extension of FindBugs [2]. To identify information needs, a collaborator and I coded transcriptions from participants' audio recordings. To identify emergent categories in the information needs, we conduced an open card sort. Our card sort was validated by two external researchers, who substantially agreed with our categorization ($\kappa = .63$ and $.\kappa = .70$, respectively). This study provides us with an initial framework to understand the vulnerability resolution process.

Phase 2 (Complete): What are developers' strategies for acquiring the information they need? We were motivated to extend our prior information needs study, because we wanted to understand how developers *answered*, or failed to answer, their questions. In this follow-up work we explored how developers acquire the information they need through strategies. To answer this second research question, we reanalyzed the data collected from the Phase 1 study to identify strategies [15].

Phase 3 (In Progress): How do existing static analysis tools support developers' information needs and strategies? During Phase 1 and Phase 2, we studied aspects of *developers' behavior* while interacting with a single security-oriented static analysis tool. To answer RQ3 we shift focus from the developer onto the tools, studying how characteristics of existing analysis tools contribute to and detract from the vulnerability resolution process.

We have conducted a heuristic walkthrough evaluation [16] of three open source security tools and plan to extend the evaluation to include commercial tools. As a result of our heuristic walkthrough evaluation so far, we have identified a list of 155 usability issues. We are also in the process of conducting interviews with security experts about their use of static analysis tools. Together, these studies will inform the design of a new static analysis tool interface (Phase 4).

Phase 4 (Proposed): How can we design tools that support more accurate and efficient resolution strategies? To answer this fourth research question I will demonstrate, through novel tool design, how we can apply our findings from the previous three research questions. Particularly, I will create a tool that explicitly supports more effective vulnerability resolution strategies. Figure 1 depicts a mockup of the tool I will build. Its interface reifies effective strategies in hierarchically structured checklists that can be executed by developers who would otherwise lack strategic knowledge.

I hypothesize that such tool will be most beneficial for novice developers, since security experts might have already internalized effective strategies. Therefore, I plan to evaluate this tool in an educational setting among students with relatively little exposure to secure software development. To measure accuracy and efficiency, we will record the number of vulnerabilities participants resolve with the new tool and how long they spend resolving each vulnerability and compare their performance against a baseline. I will triangulate these measures by also capturing usability metrics.

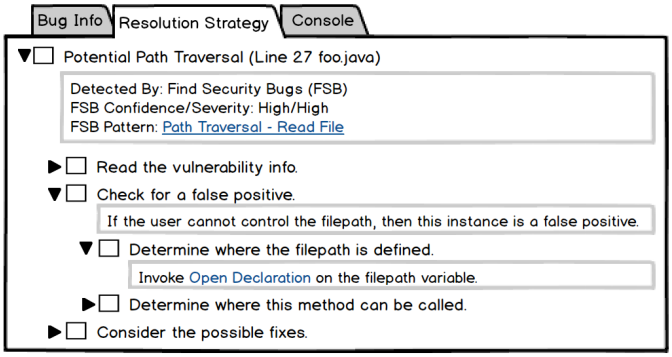

Fig. 1: A mockup of a tool that presents successful strategies.

V. Acknowledgments

I owe thanks to my advisor, Dr. Emerson Murphy-Hill, my dissertation committee, and the many collaborators who have contributed to this work. This material is based upon work supported by the National Science Foundation under grant number 1318323.

References

[1] "Coverity home page," https://scan.coverity.com/, 2018.
[2] "Findbugs," http://findbugs.sourceforge.net.
[3] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.
[4] M. Christakis and C. Bird, "What developers want and need from program analysis: An empirical study," in *IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2016. New York, NY, USA: ACM, 2016, pp. 332–343.
[5] B. Chess and G. McGraw, "Static analysis for security," *IEEE Security and Privacy*, vol. 2, no. 6, pp. 76–79, Nov. 2004.
[6] B. Johnson, R. Pandita, J. Smith, D. Ford, S. Elder, E. Murphy-Hill, S. Heckman, and C. Sadowski, "A cross-tool communication study on program analysis tool notifications," in *International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 73–84.
[7] M. Green and M. Smith, "Developers are not the enemy!: The need for usable security apis," *IEEE Security and Privacy*, vol. 14, no. 5, pp. 40–46, 2016.
[8] Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, either: A research agenda for usable security and privacy research beyond end users," in *IEEE SecDev*. IEEE, 2016, pp. 3–8.
[9] S. K. Bhavnani and B. E. John, "The strategic use of complex computer systems," *Human-Computer Interaction*, vol. 15, no. 2, pp. 107–137, Sep. 2000.
[10] T. Muske and A. Serebrenik, "Survey of approaches for handling static analysis alarms," in *IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2016, pp. 157–166.
[11] C. Sadowski, J. Van Gogh, C. Jaspan, E. Söderberg, and C. Winter, "Tricorder: Building a program analysis ecosystem," in *IEEE International Conference on Software Engineering*. IEEE Press, 2015, pp. 598–608.
[12] N. Ayewah, W. Pugh, J. D. Morgenthaler, J. Penix, and Y. Zhou, "Evaluating static analysis defect warnings on production software," in *ACM Workshop on Program Analysis for Software Tools and Engineering*. ACM, 2007, pp. 1–8.
[13] J. Smith, B. Johnson, E. Murphy-Hill, B. Chu, and H. R. Lipford, "Questions developers ask while diagnosing potential security vulnerabilities with static analysis," in *ACM International Symposium on Foundations of Software Engineering*, ser. ESEC/FSE 2015. ACM, 2015, pp. 248–259.
[14] "Find security bugs," http://h3xstream.github.io/find-sec-bugs/.
[15] J. Smith, B. Johnson, E. Murphy-Hill, B.-T. Chu, and H. Richter, "How developers diagnose potential security vulnerabilities with a static analysis tool," *IEEE Transactions on Software Engineering*, 2018.
[16] A. Sears, "Heuristic walkthroughs: Finding the problems without the noise," *Human-Computer Interaction*, vol. 9, no. 3, pp. 213–234, 1997.