Practitioner Perception of Vulnerability Discovery Strategies

Farzana Ahamed Bhuiyan*, Justin Murphy†, Patrick Morrison‡ and Akond Rahman§
*†§*Department of Computer Science, Tennessee Technological University, Cookeville, Tennessee, USA*
‡*IBM, Durham, North Carolina, USA*
Email: *fbhuiyan42@tntech.edu, †jdmurphy43@tntech.edu, ‡pjmorris@us.ibm.com, §arahman@tntech.edu

***Abstract*—The fourth industrial revolution envisions industry manufacturing systems to be software driven where mundane manufacturing tasks can be automated. As software is perceived as an integral part of this vision, discovering vulnerabilities is of paramount of importance so that manufacturing systems are secure. A categorization of vulnerability discovery strategies can inform practitioners on how to identify undiscovered vulnerabilities in software. Recently researchers have investigated and identified vulnerability discovery strategies used in open source software (OSS) projects. The efficacy of the derived strategy needs to be validated by obtaining feedback from practitioners. Such feedback can be helpful to assess if identified strategies are useful for practitioners and possible directions the derived vulnerability discovery strategies can be improvised. We survey 51 practitioners to assess if four vulnerability discovery strategies: diagnostics, malicious payload construction, misconfiguration, and pernicious execution can be used to identify undiscovered vulnerabilities. Practitioners perceive the strategies to be useful: for example, we observe 88% of the surveyed practitioners to agree that diagnostics could be used to discover vulnerabilities. Our work provides evidence of usefulness for the identified strategies.**

***Index Terms*—bug report, perception, survey, strategy, vulnerability**

I. Introduction

In January 2019, a security tester disclosed four 36 year-old vulnerabilities in the source code of secure copy protocol (SCP) [1]. SCP is an open source software (OSS), which is maintained by the OpenBSD organization [2]. The security tester constructed a malicious file to discover the vulnerabilities [3]. Despite the use of software engineering practices, such as code auditing [4] and testing [2], the vulnerabilities in SCP source code remained undiscovered. Tools that use SCP, such as, OpenSSH [1] and Putty [2], have thousands of users and would have not been negatively impacted if the vulnerabilities were discovered early. The knowledge of constructing a malicious file for vulnerability discovery might have helped OpenBSD contributors to identify the vulnerabilities earlier, instead of a security tester discovering them in 2019.

Knowledge related to vulnerability discovery strategy could be of great importance to practitioners who are involved in developing critical software systems. Understanding and applying vulnerability discovery strategy can help practitioners discover vulnerabilities early in software systems preventing malicious users from attacking software systems. Practitioners perceive software to be an integral part of the fourth industrial revolution, and latent vulnerabilities in software can provide malicious users to conduct large-scale cybersecurity attacks.

In recent work, Bhuiyan et al. [5] investigated vulnerability discovery strategies used in OSS projects. They identified four vulnerability discovery strategies: diagnostics, malicious payload construction, misconfiguration, and pernicious execution. They reported the frequency of the identified vulnerability discovery strategies used in OSS. One limitation of their work is validation with practitioners: they did not synthesize practitioners perceptions of the identified strategies. Addressing such limitation is important as practitioner agreement with the identified strategies can provide validation and evidence of usefulness for the identified strategies. Practitioner disagreements can reveal for which reasons identified strategies may not work and how research conducted by Bhuiyan et al. [5] can be improved by considering practitioner context.

We answer the following research question: ***RQ**: How do practitioners perceive the identified vulnerability discovery strategies?*

We conduct a survey with 51 practitioners to assess if practitioners perceive the identified strategies to be helpful in identifying undiscovered vulnerabilities. Our contribution is a survey of how practitioners perceive the identified vulnerability discovery strategies (Section III).

We organize the rest of the paper as follows: we describe the methodology and results respectively, in Sections II and III. We discuss our findings and implications in Section IV. We discuss related work in Section V and conclude the paper in Section VI.

II. Methodology

Bhuiyan et al. [5] did qualitative analysis on the text content of OSS bug reports available from the Mozilla organization. They identified 312 bug reports that describe what strategies security testers applied to discover the vulnerability using some filtering criteria. They used open coding [6] on the content of bug reports, including comments, attachments, and description and identified four vulnerability discovery strategies. They used five datasets to report the frequency of the identified strategies. Using the metric 'PropVuln', they quantified the proportion of vulnerabilities that were identified using each strategy.

[1]https://www.openssh.com/
[2]https://www.putty.org/

TABLE I
NAME AND DEFINITION OF EACH STRATEGIES ALONG WITH PROPORTION OF VULNERABILITIES DISCOVERED BY EACH STRATEGY. HIGHLIGHTED CELLS IN GREEN INDICATE THE MOST FREQUENTLY OCCURRING STRATEGY FOR A DATASET.

Strategy	Definition	Frequency				
		Chrome	Eclipse	Mozilla	OpenStack	PHP
Diagnostics	The strategy of inspecting software source code or software logs to discover vulnerabilities.	10.0%	62.9%	13.4%	52.1%	15.3%
Misconfig	The strategy of identifying and altering one or multiple configurations of the software to discover vulnerabilities.	0.8%	8.5%	2.7%	29.4%	17.7%
Payload	The strategy of constructing a malicious software artifact to discover a vulnerability.	86.5%	25.7%	83.2%	16.8%	89.5%
Execution	The strategy of identifying a sequence of software interactions that expose a vulnerability.	2.7%	5.7%	6.8%	4.2%	1.6%

We report the name and definition of the four identified strategies and their 'PropVuln' values for the four strategies as reported in [5] in Table I. The columns 'Diagnostics', 'Misconfig', 'Payload', and 'Execution' respectively, present the four strategies: diagnostics, misconfiguration, malicious payload construction, and pernicious execution. Except for Eclipse and OpenStack, we observe malicious payload construction to be the most frequent vulnerability discovery strategy. For Eclipse and OpenStack, the most frequently occurring strategy is diagnostics. For Eclipse and OpenStack, respectively, 62.9% and 52.1% of the vulnerabilities are discovered using diagnostics.

We answer our RQ by conducting an online survey with practitioners. In the survey, we first ask practitioners about their experience in software engineering. Next, we ask how long they have been involved in discovering vulnerabilities in software projects. Then, we ask "*We have identified four strategies to discover vulnerabilities in software by looking at bug reports from open source software. Each of these strategies is listed below. To which extent do you agree that each of our strategies could help you to identify an undiscovered vulnerability?*". We construct the survey following Kitchenham and Pfleeger's guidelines [7]: (i) use Likert scale to measure agreement levels: strongly disagree, disagree, neutral, agree, and strongly agree; (ii) add explanations related to the purpose of the study, how to conduct the survey, preservation of confidentiality, and an estimate of completion time; and (iii) conduct a pilot survey to get initial feedback. From the feedback of the pilot survey, we add an open-ended text box so that survey respondents can provide more context including feedback on the reasons they agreed or disagreed. We include the survey questionnaire in our verifiability package [8].

We deploy the survey using e-mails. We use two sources: *first*, we use a mailing list maintained by a university-affiliated organization, which is involved in vulnerability finding initiatives, such as 'capture the flag' and 'bug bounty' programs. The mailing list includes e-mail addresses of 153 practitioners. *Second*, we use 1,025 unique e-mail addresses from our collection of bug reports from Chrome, Eclipse, Openstack, and PHP. We select 300 from the set of 1,178 e-mail addresses by applying stratified sampling [9], [10]. In stratified sampling, samples are randomly selected from each group within the population so that the constructed random sample reflects the same proportionate distribution of each group [9], [10]. The sample includes 30 e-mail addresses from the university-affiliated mailing list and 270 e-mail addresses from the OSS bug reports. Our assumption is that by collecting responses from practitioners who are involved in the vulnerability discovery process either in the industry or outside the industry, we can assess the relevance of our identified strategies.

We offer a drawing of two 50 USD Amazon gift cards as an incentive for participation following Smith et al. [11]'s recommendations. We conduct the survey from January 10, 2020 to November 15, 2020 following the Internal Review Board (IRB) protocol#2242.

Limitation: Our survey is subject to internal validity as we might have used phrases in the survey that could be ambiguous to survey participants. We mitigate this limitation by conducting a pilot study with a volunteer. The volunteer reported not to face any issues related to comprehension, ambiguity, or readability.

III. RESULTS

We summarize survey results in Figure 1, where the strategies are sorted from top to bottom based on the agreement rate. The percentage of survey respondents who agreed or strongly agreed with each category is listed on the right. The agreement rate is 88%, 80%, 76%, and 75% respectively, for diagnostics, misconfiguration, malicious payload construction, and pernicious execution. Of the 300 practitioners, 51 practitioners responded to the survey. We observe the median reported experience in software engineering to be 4 years (min=1 year, max=30 years). Surveyed practitioners reported a median of 2 years of experience in vulnerability discovery (min=1 year, max=20 years).

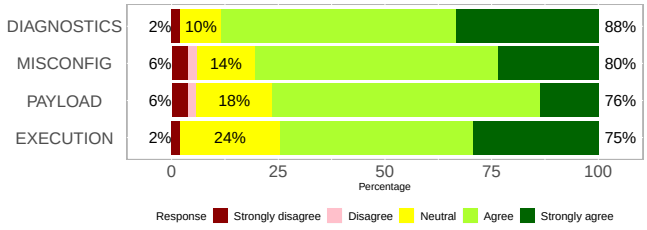

Fig. 1. Findings from survey. The most agreed upon strategy is diagnostics.

The survey respondents also provided comments on why they agreed, disagreed, or remained neutral. One survey respondent agreed with all identified strategies stating "*I have seen all of them work before and they are effective at finding vulnerabilities*". One respondent expressed a lack of experience in malicious payload construction saying "*I don't know how much impact a malicious payload has, but it sounds like it's important so I put neutral*". One respondent disagreed with malicious payload construction stating the strategy to be"*situationally beneficial*". The respondent perceived the success of finding vulnerabilities using malicious payload construction to be dependent on developer mistakes in source code. Another respondent suggested that other strategies may exist, which are not included in our list: "*There may be more strategies added as we go on identifying more and more CVEs*".

IV. Discussion

Despite disagreements we observe an overall agreement from the surveyed practitioners for all the identified strategies. We observe an agreement rate of $>= 75\%$ for all the identified strategies. Our survey results provide further practitioner validation to the strategies identified by Bhuiyan et al. [5]. Results presented in Figure 1 provide evidence that practitioners who are interested in early discovery of vulnerabilities can leverage the four strategies to identify latent vulnerabilities.

Srikanth and Menzies [12] stated "*Documenting developer beliefs should be the start, not the end, of software engineering research. Once prevalent beliefs are found, they should be checked against real-world data*", suggesting software engineering researchers to complement practitioner survey data with software repository data. We have complemented practitioner survey results with strategy frequency results mined from OSS bug reports in Bhuiyan et al. [5]. We observe congruence: diagnostics is the most agreed-upon strategy, and also the most frequently occurring strategy for Eclipse and OpenStack.

We also observe practitioner responses to be rooted in personal beliefs and experiences, which has previously been documented for software quality research [13]. For example, in Section III we observe a practitioner to express skepticism about malicious payload construction. Skeptic practitioners might benefit from more details such as definitions, examples, and subcategories related to malicious payload construction and how frequently the strategy is used in software projects. Practitioners often prefer to learn through the experiences of their peers i.e., other practitioners working in the same domain [14]–[16]. We advocate practitioners who are inexperienced in vulnerability discovery to learn from the presented results in our paper as survey results how the strategies are validated and our results are a synthesis of strategies used by security testers.

Ethical Disclosure of Vulnerabilities: Disclosure of software vulnerabilities have a negative impact on a software product's economic value [17]. Regardless of what strategy is being used, if a new vulnerability is discovered, then practitioners must follow the best practices when reporting the vulnerability. Practitioners may learn from existing vulnerability reporting philosophies used by IT organizations [18]–[20] that include (i) mutual agreement on vulnerability disclosure dates, and (ii) not attempting to access project or user data that may result from the vulnerability exploit.

V. Related Work

Our paper is related to software engineering research that has used bug reports for vulnerability analysis and has focused on security testers. Researchers have used OSS bug reports to characterize and identify software vulnerabilities. Shin and Williams [21] mined Mozilla bug reports to construct fault prediction models and reported that the models could be used to predict vulnerable source code files automatically. In another work, Zhe et al. [22] mined Mozilla bug reports to construct vulnerability datasets, and applied active learning to improve vulnerability detection efficiency. Researchers have also applied qualitative analysis to characterize software vulnerabilities. Linares-Vasquez et al. [23] applied qualitative analysis on Android bug reports and identified the 'Linux Kernel' component of Android to contain 46% of the studied 634 vulnerabilities. Using bug reports Santos et al. [24] identified 44 root causes for architectural vulnerabilities in three software projects. Prior research has also focused on security testers i.e., practitioners who discover software vulnerabilities. Smith et al. [25] observed security testers to face barriers when using static analysis tools for vulnerability discovery. Rahman and Williams [26] studied the knowledge needs of inexperienced security testers and observed limited availability of resources to discover software vulnerabilities. Ceccato et al. [27] modeled behaviors of security testers on how they diagnose source code to find vulnerabilities. Munaiah et al. [28] studied system call logs from a Capture the Flag competition and observed security testers to perform a set of computational activities in chronological order. Votipka et al. [29] interviewed software testers and security testers and reported that even though software and security testers perform the same computational and cognitive processes, the outcome is different for software testers compared to that of security testers. Fang and Hafiz [30] studied vulnerability reporting practices using a survey with 58 security testers and observed security testers to not collaborate with software vendors when reporting buffer overflow vulnerabilities. The research paper closest in spirit to our paper is research conducted by Bhuiyan et al. [5] who investigated vulnerability discovery strategies used in OSS projects and reported four vulnerability discovery strategies along with their frequency in OSS projects.

Our discussion above highlights a lack of research that studies practitioner perception on research related to vulnerability discovery strategies in the OSS domain. We address this research gap by conducting a survey of how practitioners perceive vulnerability discovery strategies.

VI. Conclusion

Bringing the fourth industrial revolution into reality will require secure development of software that is being used in industrial automation systems. Software is an integral part of the the fourth industrial revolution and latent vulnerabilities in software can provide malicious users to conduct cybersecurity attacks. While researchers have derived strategies to discover vulnerabilities, how practitioners perceive the identified strategies remain under explored. We conduct a survey with 51 practitioners to assess if previously identified four strategies have relevance to practitioners. From our survey, the most and least agreed upon strategy is, respectively, diagnostics and pernicious execution. We hope our paper will facilitate further research in the domain of vulnerability discovery.

Acknowledgement

We thank the PASER group at Tennessee Technological University for their valuable feedback. We also thank the anonymous practitioners for participating in our survey. The research was partially funded by the National Science Foundation (NSF) NSF award # 2026869.

References

[1] C. Cimpanu, "Scp implementations impacted by 36-years-old security flaws," https://www.zdnet.com/article/scp-implementations-impacted-by-36-years-old-security-flaws/, 2019, [Online; accessed 14-Jan-2020].

[2] OpenBSD, "Openbsd ports - testing guide," https://www.openbsd.org/faq/ports/testing.html, 2020, [Online; accessed 21-Feb-2020].

[3] H. Sintonen, "https://sintonen.fi/advisories/scp-client-multiple-vulnerabilities.txt," https://sintonen.fi/advisories/scp-client-multiple-vulnerabilities.txt, 2019, [Online; accessed 27-Jan-2020].

[4] OpenBSD, "Openbsd:security," https://www.openbsd.org/security.html, 2020, [Online; accessed 21-Feb-2020].

[5] F. A. Bhuiyan, A. Rahman, and P. Morrison, "Vulnerability discovery strategies used in software projects," in *35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW'20)*. IEEE, 2020.

[6] J. Saldana, *The coding manual for qualitative researchers*. Sage, 2015.

[7] B. A. Kitchenham and S. L. Pfleeger, *Personal Opinion Surveys*. London: Springer London, 2008, pp. 63–92. [Online]. Available: https://doi.org/10.1007/978-1-84800-044-5_3

[8] A. Authors, "Verifiability package for paper," https://figshare.com/s/eaac5aeea283239f2c56, 2020, [Online; accessed 05-Mar-2020].

[9] M. N. Marshall, "Sampling for qualitative research," *Family Practice*, vol. 13, no. 6, pp. 522–526, 12 1996. [Online]. Available: https://doi.org/10.1093/fampra/13.6.522

[10] M. D. Kaplowitz, T. D. Hadlock, and R. Levine, "A Comparison of Web and Mail Survey Response Rates," *Public Opinion Quarterly*, vol. 68, no. 1, pp. 94–101, 03 2004. [Online]. Available: https://doi.org/10.1093/poq/nfh006

[11] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and T. Zimmermann, "Improving developer participation rates in surveys," in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 2013, pp. 89–92.

[12] S. N. C. and T. Menzies, "Assessing developer beliefs: A reply to "perceptions, expectations, and challenges in defect prediction"," *CoRR*, vol. abs/1904.05794, 2019. [Online]. Available: http://arxiv.org/abs/1904.05794

[13] P. Devanbu, T. Zimmermann, and C. Bird, "Belief and evidence in empirical software engineering," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. ACM, 2016, pp. 108–119. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884812

[14] G. A. Moore and R. McKenna, "Crossing the chasm," 1999.

[15] A. A. U. Rahman, E. Helms, L. Williams, and C. Parnin, "Synthesizing continuous deployment practices used in software development," in *Proceedings of the 2015 Agile Conference*, ser. AGILE '15. USA: IEEE Computer Society, 2015, p. 1–10. [Online]. Available: https://doi.org/10.1109/Agile.2015.12

[16] E. Murphy-Hill and G. C. Murphy, "Peer interaction effectively, yet infrequently, enables programmers to discover new tools," in *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, ser. CSCW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 405–414. [Online]. Available: https://doi.org/10.1145/1958824.1958888

[17] R. Telang and S. Wattal, "An empirical analysis of the impact of software vulnerability announcements on firm stock price," *IEEE Transactions on Software Engineering*, vol. 33, no. 8, pp. 544–557, Aug 2007.

[18] Lookout, "Responsible disclosure," https://www.lookout.com/legal/responsible-disclosure, 2020, [Online; accessed 19-Feb-2020].

[19] H. One, "Vulnerability disclosure guidelines," https://www.hackerone.com/disclosure-guidelines, 2020, [Online; accessed 02-Feb-2020].

[20] G. A. Security, "How google handles security vulnerabilities," https://www.google.com/about/appsecurity/, 2020, [Online; accessed 01-Feb-2020].

[21] Y. Shin and L. Williams, "Can traditional fault prediction models be used for vulnerability prediction?" *Empirical Software Engineering*, vol. 18, no. 1, pp. 25–59, Feb 2013. [Online]. Available: https://doi.org/10.1007/s10664-011-9190-8

[22] Z. Yu, C. Theisen, L. Williams, and T. Menzies, "Improving vulnerability inspection efficiency using active learning," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[23] M. Linares-Vasquez, G. Bavota, and C. Escobar-Velasquez, "An empirical study on android-related vulnerabilities," in *Proceedings of the 14th International Conference on Mining Software Repositories*, ser. MSR '17. IEEE Press, 2017, p. 2–13. [Online]. Available: https://doi.org/10.1109/MSR.2017.60

[24] J. C. Santos, K. Tarrit, A. Sejfia, M. Mirakhorli, and M. Galster, "An empirical study of tactical vulnerabilities," *Journal of Systems and Software*, vol. 149, pp. 263 – 284, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121218302322

[25] 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 *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, 2015, p. 248–259. [Online]. Available: https://doi.org/10.1145/2786805.2786812

[26] A. Rahman and L. Williams, "A bird's eye view of knowledge needs related to penetration testing," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, ser. HotSoS '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3314058.3317294

[27] M. Ceccato, P. Tonella, C. Basile, B. Coppens, B. De Sutter, P. Falcarin, and M. Torchiano, "How professional hackers understand protected code while performing attack tasks," in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, May 2017, pp. 154–164.

[28] N. Munaiah, A. Rahman, J. Pelletier, L. Williams, and A. Meneely, "Characterizing attacker behavior in a cybersecurity penetration testing competition," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Sep. 2019, pp. 1–6.

[29] D. Votipka, R. Stevens, E. Redmiles, J. Hu, and M. Mazurek, "Hackers vs. testers: A comparison of software vulnerability discovery processes," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 374–391.

[30] M. Fang and M. Hafiz, "Discovering buffer overflow vulnerabilities in the wild: An empirical study," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2652524.2652533