Towards More Effective Responsible Disclosure for Vulnerability Research

Weiheng Bai
University of Minnesota
bai00093@umn.edu

Qiushi Wu
University of Minnesota
wu000273@umn.edu

***Abstract*—Vulnerability research is vital to mitigating cyber-attacks, which tries to devise new approaches to discover new vulnerabilities. As an ethical research guideline, researchers are expected to report the found vulnerabilities to the corresponding vendors before disclosing them (e.g., publishing a paper), which is known as the responsible-disclosure process. Undoubtedly, the intention of responsible disclosure is to help improve the security of software. We observe that the current responsible disclosure may not be as effective as expected. In particular, reports can be significantly delayed or completely ignored. Reports for security-critical vulnerabilities are often publicly disclosed, which can potentially be abused by attackers.**

In this work, we plan to study the effectiveness of the existing responsible disclosure. Two major questions we aim to answer are: (1) Are security-critical bug reports commonly disclosed publicly in the first place? (2) What factors of a bug report contribute to delaying or ignoring? By answering the questions, we aim to provide insights into how to improve the quality of bug reports and the effectiveness of responsible disclosure. In this paper, we present our preliminary results of this work. We take the Linux reports and patch history as an example. We found that at least in Linux, most security bugs are publicly disclosed before they are fixed, and that factors such as length of reports, author experience, and author affiliations have an impact on the delay of patching. In the end, we also present our plans for future work.

I. Introduction

Vulnerabilities, or security bugs, have been a major threat to cyber-security. We witnessed critical security breaches resulted from vulnerabilities. As vulnerabilities essentially stem from logic bugs or design flaws, they are hard to prevent. Therefore, finding and fixing vulnerabilities have become a practical and common security strategy that is widely adopted by software vendors. For example, almost all major software vendors, such as Google and Microsoft, have been running vulnerability-disclosure programs or bounty programs.

Whenever a researcher finds a vulnerability or just a bug, the researcher is expected to report it to the related parties. This is called *responsible disclosure*, a common ethical research guideline. All major security conferences have explicitly enforced this policy. While the responsible-disclosure process is intended to help maintainers proactively identify and fix vulnerabilities, and thus to improve the security, we observe that it may not be as effective as expected. In particular, we have observed two common issues with responsible disclosure. First, many security-critical bugs are publicly disclosed in the first place before they are fixed. While this can be a result of that the security impacts of a bug are unknown when reporting, we also found many cases in which the reporters identified the security impacts and still publicly reported them. Such reports may be abused by attackers for exploitation, which undermines the intention of responsible disclosure. Second, many reports are of low quality; as a result, reports are often completely ignored by maintainers, or their acceptance is significantly delayed, which increases the length of the unpatched period.

In this work, we aim to study the effectiveness of existing responsible disclosure. We are concerned with two major questions. (1) Is it common that security-critical bug reports are disclosed publicly in the first place? (2) What factors of a bug report contribute to delaying or ignoring? By answering the questions, we aim to provide insights into how to improve the quality of bug reports and the effectiveness of the responsible disclosure.

In this paper, we present our preliminary results of the work. We select Linux bug reports (including the ones ignored, discussed, and accepted) and its patch history as an example. We perform a quantitative analysis on the reports to derive statistical results. Contrary to the finding in [4], which claims that most vulnerabilities are fixed before they are publicly disclosed, we however found that most vulnerabilities in Linux are publicly disclosed before they are fixed. Such disclosure approaches may raise ethical concerns, which violate the responsible-disclosure process, and moreover, adversaries may exploit such vulnerabilities against the affected programs. We also found that factors such as length of reports, author experience, and author affiliations have an impact on the delay of patching.

As a future work, we plan to collect bug reports for other projects through platforms such as GitHub.com and Bugzilla. We also plan to analyze the reports from more aspects such as factors leading to regression bugs and to develop automated solutions to assist both reporters and maintainers for identifying security impacts of bugs. Through this work, we hope to provide guidelines and tools for a more effective responsible-disclosure process.

Workshop on Ethics in Computer Security (EthiCS) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-85-1
https://dx.doi.org/10.14722/ethics.2023.235691
www.ndss-symposium.org

II. Approach Overview

We aim to answer the two research questions through a quantitative study based on existing bug reports. Therefore, our method is to first collect necessary data from multiple sources including the bug reports (emails on public mailinglists), Git history, and the CVE database. We then check the timestamps, propose a hypothesis of potential factors contributing to delays and ignores, and finally perform a correlation analysis between the factors and delays. We show the workflow of our method in Figure 1.

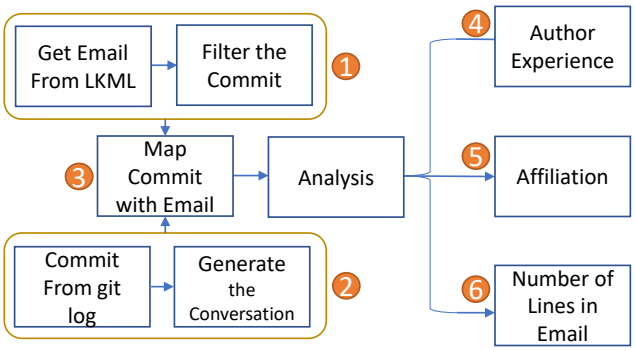

Fig. 1: The workflow of our study method.

III. Collecting the Data

First, we collect the Git commits (i.e., accepted bugs reports) from Linux `Git` log and filter the bug-fixing commits. Meanwhile, we collect the bug-reporting emails sent to Linux through the public mailinglist, `LKML`. These are original bug reports that can be ignored or accepted into Linux. After that, we need to cluster emails into threads if the emails are for discussing and revising the same bug. We collect the Linux commits with `Git` head 52d543b5497cf. Also, we collect the emails from LKML from 01/01/2017 to 03/21/2022.

After the first two steps, we have bug-fixing commits from the `Git` log and email threads for bugs. We then map the the commits with their corresponding email threads.

Collecting Git commits. We first collect all the commits from the `Git` log, and then we need to filter out the irrelevant commits since we want to have the bug-fixing commits. So, we set up some irrelevant words as the keywords we want to filter out, such as, `'typo'`, `'grammar'`, `'spelling'`, etc. Such commits are not for bugs but coding styles. We present a summary of the commits in Table I.

Year	Total Commits	Fix-related Commits
2017	80,850	9,563
2018	80,161	9,406
2019	82,915	10,342
2020	90,329	11,164
2021	85,156	11,133
2022	13,518	1,716
Total	1,085,249	53,326

TABLE I: The Distribution of collected commits

We aim to collect as much information as possible. For each commit, we collect commit hash, submit date, author, author email, commit date, committer, committer email, commit summary, commit size and commit message.

Crawling bug-reporting emails. This part is shown on the Figure 1 ①. In this part, we collect the emails from https://lkml.org/lkml/. The date range is from 01/01/2017 to 03/21/2022. We then apply the same strategy to filter the bug-fixing emails. We present a summary of the crawled emails in Table II.

Year	Emails	Fix-related Emails	Conversations
2017	274,798	30,169	16,636
2018	312,810	35,847	22,135
2019	357,936	51,136	32,062
2020	423,133	57,574	35,671
2021	407,657	56,436	34,930
2022	91,023	10,316	6,122
Total	1,867,357	241,478	147,556

TABLE II: The distribution of bug-fixing emails

Mapping commits and emails. The next step is to match the commits with the emails. First, we need to check whether a commit ID exists inside the an email conversation. Through this check, we could match most of the commits with email conversations. For remaining cases, we need to check the authors of the commits, authored time of the commits, which will let us have a set of email conversations matched with the commits. Then, we compare the patches inside the commits with the patches inside the conversations. Based on the rules, we are able to prepare a dataset with matched commits and email conversations.

With the collected data, we can then conduct a quantitative study to answer the two research questions, which is presented in the next section.

IV. Preliminary Results

In this section, we present preliminary results of our quantitative study.

A. Unethical Public Vulnerability Reports

As a part of the responsible disclosure process, security-critical bugs should be reported through private channels. Therefore, we first study if publicly disclosing security-critical bugs is a common problem. To this end, we check if CVE-assigned vulnerabilities were disclosed before they are fixed. In particular, we select 100 latest Linux-related vulnerabilities with CVE numbers assigned. We then manually check the content of each case and identify the corresponding commit ID in the Git history of Linux. The commit ID allows us to identify the corresponding fix and original bug report. This way, we can tell when the vulnerabilities were disclosed (reported) and fixed. The results are surprising to us, as they show that for all these 100 vulnerabilities, they were all publicly reported before they were fixed. If attackers have incentive to abuse the bug reports, they can simply monitor the public bug reports and take advantage of them. Such a responsible disclosure can thus turn in to a harmful process. In fact, Linux provides a private channel (`security@linuxfoundation.org`). Unfortunately, Linux reporters mostly did not use the private channel for the responsible disclosure.

B. Factors Contributing to Delays

In this section, we first perform an analysis to understand the delays and their distribution, and then conduct a study to understand the correlation between the factors and the delays.

Distribution of delays. We define *delay* as the time gap between the time of the original bug report (the email) and the time of patch merging (commit). Since memory-corruption bugs such as buffer overflow and user-after-free are generally security-critical, we take it as the target type of bugs. By checking the keywords, we finally collect 1,818 such bugs and measure the distribution of their delays. The result is shown in Figure 2. We could see that only 22.05% of the cases have a delay of less than one day, but 87.95% of the cases have more than 1 day to be expose to the public. 7.73% of the cases even have a delay of longer than one month, which may give attackers time to abuse the reports.

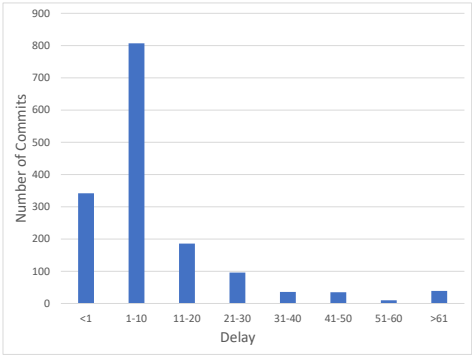

Fig. 2: Distribution of delays in days

We next take the delays as the metrics to measure factors contributing to the delays. As a preliminary study, we hypothesize that there are at least three factors that have a correlation with the delays: *length of the report*, *author experience*, and *author affiliation*. Note that in the following study, to have a more robust statistical analysis, we use a larger dataset that contains 37,971 commits.

Author experience. We define *author experience* based on how many commits (i.e., accepted bug reports or patches) has the author contributed to. Such information can be obtained from the `Git` log. we hypothesize that an author with more experience tends to write higher-quality bug reports that will have a shorter delay to be accepted. The reason why we choose the geo-mean to represent the delay time is that fluctuation in sampling will not affect the geo-mean result. On the opposite, if we try to calculate the arithmetic mean, when there is a delay time extremely larger than other delay time, it will cause the arithmetic mean increasing significantly. In addition, to save the computing power and computing time, we use log() to narrow down the intermediate value during the process of geo-mean calculation. To confirm it, we perform an analysis against the correlation between experience and delays. Figure 3 shows the results.

The results are surprising to us—they are opposite to our hypothesis. From Figure 3 we see that the authors with the number of submission under 65 (i.e., less experienced authors

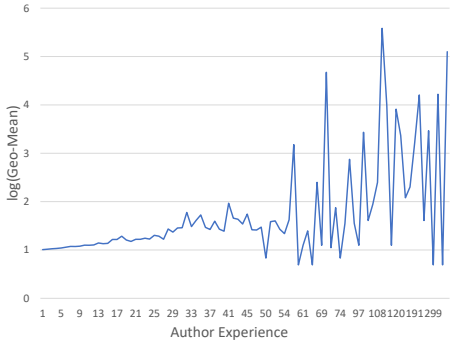

Fig. 3: Correlation: Author experience & Delays

based on our definition) have relatively smaller delays in general. On the other hand, more experienced authors with more than 65 confirmed commits instead have relatively longer delays. Such an unexpected finding deserves further investigation. We will explore potential reasons as our future work.

Motivated by this finding, we further performed a statistical analysis on acceptance rate of bug reports from experienced and less experienced authors. The results are however very different—experienced authors have a much higher acceptance rate, which is expected.

Author affiliations. We hypothesize that author affiliation may also have an impact on the delays. To confirm this, we first classify affiliations in to six categories: personal, education, Linux department of companies, companies, Linux-related organizations, and other organizations. We use the mapping data to extract the information based on email address and then categorize the data into the categories. We present the results in Table III, which shows that, bug reports from educational organization (e.g., universities) tend to have a much longer delay (almost twice as the average). This is a bit surprising to us, so we looked into it. We found that since authors from the education affiliation are often students; they may not have much experience on submitting a patch report. Such reports tend to have some basic mistakes. For example, every email sent to Linux maintainer should cc `linux-kernel@vger.kernel.org`. But some of them forgot to cc this email, so the maintainer have to ask the reporter do that, which increase the delay time. On the other hand, bug reports from Linux department and companies have the smaller delay, which is consistent to our hypothesis.

Days (Geo-Mean)	E	LRO	O	P	LDC	C
	3.65	3.25	2.76	2.73	2.72	2.72

TABLE III: Correlation: Affiliation & Delays. E: Education; LRO: Linux-related organizations; O: Other organizations; LDC: Linux department of companies; P: personal; C: other companies.

Length of reports. We define *length* of a report as the number of lines of textual description. We present the results in Figure 4. We can see that the correlation between the length and delays is also clear. Generally, bug reports with a lengthy description tend to have a longer delay. There are several reasons. First, a lengthy report often covers the fixes for multiple bugs. Second, the bug can be complicated and requires more review efforts.

Such a result implies that, to avoid delays, reporters should write concise description for bugs separately.

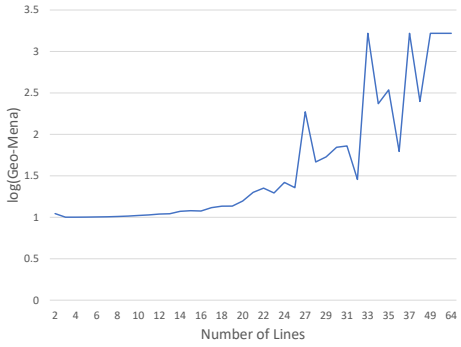

Fig. 4: Correlation: Number of lines & Delays

V. Discussion

Ethical concerns and Insights. Not properly using the private reporting channel may violate the ethical requirement of the responsible-disclosure process, e.g., Linux vulnerabilities are often publicly reported. Our results also indicate that there is typically a large time gap between the bug public disclosure to the actual patched date. Attackers can abuse both issues to cause harms. In these scenarios responsible disclosure can even turn into a harmful process. As an immediate remediation, we need to encourage reporters to use private channel for reporting vulnerabilities. Furthermore, due to the security impact of bugs being unclear, both reporter and the maintainer may not be aware of the security impact of the bug. Thus, it is essential to have an automatic tool for inferencing the security of bugs based on their patches.

In the relationship between author affiliation and delay time, companies enjoy a shorter delay. Educational organizations suffer from a longer delay. We believe a training for students on how to improve the report quality would help. Last, a concise and per-case report is preferred and can avoid longer delays.

Future work. Improving the effectiveness of responsible disclosure is an important topic that deserves further studies. We envision multiple directions for the future work. First, we plan to identify more factors and study their impacts on delays. In addition to delays, we plan to also use ignore rate (percentage of bug reports being ignored) as a metric to measure the factors. Second, beyond Linux, we plan to study more popular open-source projects through other bug-tracking platforms such as GitHub and Bugzilla. We hope to cover different categories of open-source projects, including OS, browsers, web servers, script engine, mobile/IoT apps, and libraries. Third, as a longer-term work, we plan to develop automated tools that help reporters write higher-quality reports and help maintainers proactively identify potential security impacts of a reported bug.

VI. Related work

Some previous works also performed the empirical studies for the vulnerabilities and their patches. Li et al. [3] did a large-scale patches analysis, which analyzed more than 4,000 bug fixes. It shows that for open-source projects, attackers typically have weeks to months to attack the not patched system before the patch distribution. DiffCVSS [6] shows that vulnerabilities can cause different security impacts in different contexts, which indicates that early disclosure of a trivial vulnerability may still cause severe security impacts in other dependencies. Piantadosi at al. [4] did an empirical study on the vulnerabilities of Apache server and Apache tomcat. Specifically, they claim that most vulnerabilities are fixed before they are publicly disclosed. This is the opposite to our findings. Such a conclusion does not hold at least for Linux bugs. Ding et al. [2] showed an empirical study for bugs identified by OSS-Fuzz. They claim that people rarely file CVEs for security bugs identified by OSS-Fuzz due to lacking a security background. This can also confirm that security bugs are publicly disclosed in the first place. Arora et al. [1] did an empirical study on the impacts of vulnerability disclosure. They show that vulnerability disclosure would increase the frequency of attacks. However, such attacks will decrease over time. Ramsauer et al. [5] show that even the code that arises from secret channels can also be obtained through reverse engineering methods by attackers and further exploit the systems.

VII. Conclusion

While responsible disclosure has been widely enforced as an ethical research requirement, it may not be as effective as expected. Security-critical bugs are commonly reported publicly, which can be abused by attackers. On the other hand, we observe significant delays and ignores in accepting the bug reports. In this preliminary study, we identified some potential factors (e.g., author experience, author affiliation, and report length) related to the delays and performed a quantitative study against their correlation. In the future, we plan to continue the study by looking into more open-source projects and developing automated tools for both reporters and maintainers to realize an more effective responsible-disclosure process.

References

[1] A. Arora, R. Krishnan, A. Nandkumar, R. Telang, and Y. Yang. Impact of vulnerability disclosure and patch availability-an empirical analysis. In *Third Workshop on the Economics of Information Security*, volume 24, pages 1268–1287, 2004.

[2] Z. Y. Ding and C. Le Goues. An empirical study of oss-fuzz bugs. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 131–142. IEEE, 2021.

[3] F. Li and V. Paxson. A large-scale empirical study of security patches. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2201–2215, 2017.

[4] V. Piantadosi, S. Scalabrino, and R. Oliveto. Fixing of security vulnerabilities in open source projects: A case study of apache http server and apache tomcat. In *2019 12th IEEE Conference on software testing, validation and verification (ICST)*, pages 68–78. IEEE, 2019.

[5] R. Ramsauer, L. Bulwahn, D. Lohmann, and W. Mauerer. The sound of silence: Mining security vulnerabilities from secret integration channels in open-source projects. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 147–157, 2020.

[6] Q. Wu, Y. Xiao, X. Liao, and K. Lu. OS-Aware vulnerability prioritization via differential severity analysis. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 395–412, Boston, MA, Aug. 2022. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/wu-qiushi.