Security Versus Performance Bugs: How Bugs are Handled in the Chromium Project

Amrit Rajbhandari
University of New Orleans, USA
arajbhan@uno.edu

Minhaz F. Zibran
Idaho State University, USA
MinhazZibran@isu.edu

Farjana Z. Eishita
Idaho State University, USA
FarjanaEishita@isu.edu

***Abstract*—Bug fixing is a very important activity of software maintenance. Given the recent highlight on security and privacy, one may expect that the software vendors would give security bugs a higher priority in their bug fixing process. In this paper, we present an exploratory study of different categories (i.e., security, performance, and other) of bugs in the maintenance of the Chromium browser. In particular, we study the phenomena such as how much time is spent in bug triage, how fast different types of bugs are fixed, variations of developers' experiences who fix those bugs, and show often those fixed bugs are reopened.**

We find that the performance bugs are triaged and fixed faster. Security bugs, for fixing, are assigned to more experienced developers. All categories of bugs are almost equally reopened once closed.

***Index Terms*—Security, Vulnerability, Performance, Bug, Defect, Chromium, Browser, Empirical Study**

I. Introduction

Technology today rarely exists without a software component or interface as more and more systems are being software operated. But, incidents of software failures and vulnerabilities repeatedly make news headlines since the emergence of software to date. In 2017 alone, almost *$1.7 trillion* in assets, and *3.7 billion* people were affected by software failures [5], [14]. This is fact despite bug-fixing activities consume a vast amount of total expenses in software maintenance [3] while nearly 80% of software cost is spent in maintenance [8].

With the increasing concern of cybersecurity and the expanding ubiquity of web applications and web services, the security and performance of web browsers demand more attention than before. Thus, in the bug fixing activities, at least for the web browser projects, the security bugs can be expected to be treated with higher priority. But the question remains, is it the case in practice? Program traits such as security and performance are often not considered within the functional requirements. Thus, software bugs related to such non-functional qualities might be given lower priorities as keeping the software product feature-rich with new functionalities often remains the main goal in practice.

In the light of these two possibilities and more, in this paper, we present an empirical study of the bug fixing activities in the maintenance of the Chromium browser project. We compare and contrast among the security bugs, performance bugs, and other bugs. For these different categories of bugs, we, in particular, examine different phenomena such as how much time is spent in bug triage, how fast the bugs are fixed, how experience are the developers who fix those bugs, and how often bugs are reopened after they are fixed, verified, and closed.

Thus, with respect to the three categories (i.e., security, performance, other) of Chromium bugs, we formulate the following research questions to address in this paper.

RQ1: *Which category of bugs are fixed faster and which category of bugs take longer time to be fixed?*
An answer to this research question will inform us about which bugs in the Chromium project is fixed with higher priorities.

RQ2: *Which category of bugs are fixed by more experienced developers?*
An answer to this research question will give us an idea about which categories of bugs are consuming the most experienced developers.

RQ3: *Which operating system (OS) platforms trigger majority of the bugs and which category of bugs triggered on which platform take longer to be fixed?*
An answer to this research question will advance our understanding of which OS platforms are more problematic for the Chromium browser and thus may require extra attention by the developers.

The insights drawn from the answers to all these research questions collectively will help the Chromium developers in restructuring their bug-fixing strategies (e.g., bug prioritization, bug assignment, test emphasis) to better fit their goal. The findings will also be useful for similar software projects in defining strategies for defect management.

The rest of the paper is organized as follows. Section II describes the methodology of our study. The findings derived from qualitative and quantitative analyses are presented in Section III. In Section IV, we discuss the possible threats to the validity of the results. Section V discusses other studies related to this work. Finally, Section VI concludes the paper.

II. Methodology

We address the research questions by studying Chromium bug reports. As mentioned before, the methodology of our study is very similar to that of the work of Zaman et al. [15]. The procedural steps of our work are briefly summarized in Figure 1.

978-1-6654-8350-6/22/$31.00 ©2022 IEEE
SERA 2022, May 25-27, 2022, Las Vegas, USA

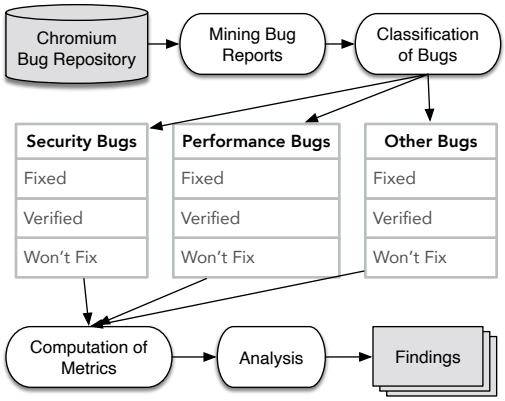

Fig. 1. Procedural steps of our empirical study

A. Collection of Bug Reports

For this study on bugs, we obtain bug reports from the Chromium bug repository [2]. On this bug repository's 'search' field, we use the following search criteria for selecting a subset of bugs from "all issues":

```
status:Fixed,Verified,WontFix
opened>2012/5/22 opened<2020/11/29
```

With these criteria, we are able to obtain only those bug reports, which satisfy the following two specifications: (a) the bug report posted/opened between *May 21, 2012*[1] and *Nov 28, 2020*, (b) the bug reports having official status 'Fixed', 'Verified', or 'WontFix' (i.e., Won't Fix). The Chromium project bug reporting guideline [1] describes these three categories of closed bugs as follows:

- *Fixed:* "Fixed."
- *Verified:* "The fix has been verified by test or by the original reporter."
- *WontFix:* "Covers all the reasons we chose to close the bug without taking action (can't repro, working as intended, obsolete)."

We collect total 189,668 closed bug reports among which 59,555 are 'Fixed' bugs, 65,864 are 'Verified' bugs, and the rest 64,249 are 'WontFix' bugs.

Figure 2 presents an example search result containing bug posts for four bugs. In Chromium bug repository, some bugs are labeled with some tags ('AllLabels' column in Figure 2). Some bug reports also include information about the operating system (OS) platforms on which the bugs are triggered ('OS' column in Figure 2). For example, the bug with ID 1153450 is triggered on the Android OS only. The bug 1153437 is triggered on Linux, Windows, and Mac OS only, while the bug 129102 is triggered on all OS platforms.

[1]It may appear that the above search criteria would include bugs opened after May *22*, 2012. But actually, with the first date in the search criteria, we obtain bugs opened on May *21*, 2012 or afterwards. We did not investigate the reason.

B. Classification of Bugs

As mentioned before, for the bugs reported in the Chromium bug repository, there is a keyword-based labelling/tagging mechanism ('AllLabels' column in Figure 2), which are used in documenting different aspects of the bugs.

TABLE I
CRITERIA FOR BUG CLASSIFICATION

For Security Bugs	For Performance Bugs
• Review-Security • Security_Impact-Stable • A11ySecurityUIQ12020 • Security • Security_Severity-Low • Security_Impact-None • Security_Severity-Medium • Security_severity-None • Team-Security-UX • Hotlist-WebAppInstalls-Security • Via-Wizard-Security • Security_Impact-Head	• CrOSFilesCategory-Performance • Performance-Loading • Performance-Size • Performance-Power • Performance-Sheriff • Performance-Startup • Performance-Battery • Performance-Memory • Performance-Tool • ntp-epic-performance • Performance-Sheriff-Feedback • Performance • Performance-Responsiveness • Test-Performance • Performance-Browser • Performance-Media • Performance-Sheriff-V8 • Performance-Sheriff-BotHealth

TABLE II
BUGS OF DIFFERENT CATEGORIES AND STATUS

Bug Category	Fix Status	Count
Security Bugs	Fixed	1,934
	Verified	4,177
	Won't Fix	3,562
	All	9,673
Performance Bugs	Fixed	2,410
	Verified	1,146
	Won't Fix	1,2481
	All	16,037
Other Bugs	Fixed	55,211
	Verified	60,541
	Won't Fix	48,206
	All	163,958
All bugs across all three categories		189,668

For classifying the bugs into one of the three categories (i.e., security, performance, or other), we use the keywords presented in Table I. If any of the terms in the left column of the table is found in the title (i.e., "Summary+Labels" column in Figure 2), description, or 'AllLabels' tags of a bug report, the corresponding bug is classified as a security bug. A particular bug is classified as a performance bug, when any of the term in the right column is found in the title, description, or 'AllLabels' tags of the corresponding bug report. For example, the bug with id 1153449 in Figure 2 is identified as a performance bug as the 'AllLabels' tags for this bug include '`performance-sheriff`'. A bug is classified as "other bugs" when it is classified neither as a security bug nor a performance bug. In Table II, we present the number

ID	Status	Summary + Labels	Opened	AllLabels	OS
1153450	Fixed	Incognito-Mode Changes to Site Settings Are Reflected in Non-Incognito	Nov 28, 2020	----	Android
1153449	WontFix	[V8 Perf Sheriff]: 5 regressions in jetstream2	Nov 28, 2020	Performance-Sheriff-V8, Chromeperf-Auto-Triaged	----
1153437	Verified	CHECK failure: false. Passed-in parent in ax_tree_serializer.h Reproducible ClusterFuzz	Nov 28, 2020	Stability-Crash, Reproducible, Stability-Memory-AddressSanitizer, ClusterFuzz, ClusterFuzz-Verified, Stability-AFL, Test-Predator-Wrong, Test-Predator-Auto-Components, M-89, Team-Accessibility	Linux, Windows, Mac
129102	Fixed	testNoMouseLockInBrowserFS broken by r138150 pyauto_tests	May 21, 2012	Hotlist-ImportantForGames, pyauto_tests, Restrict-AddIssueComment-Commit	All

Fig. 2. Examples of Chromium Bug Reports

of all these three categories of bugs along with their fix status (i.e., 'Fixed', 'Verified' or 'WontFix').

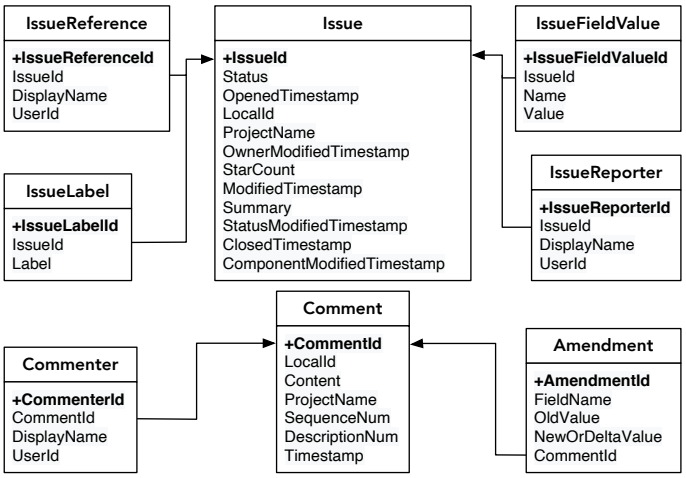

Fig. 3. ER Diagram of the Database Presenting the Information Captured

Each of the bug reports are then further parsed to extract detailed information, which are then stored in a database for further analysis. Figure 3 shows the Entity-Relationship (ER) diagram that presents the database schema and the attributes for persistence. Primary keys for each database relation/table is identified with bold font and a '+' symbol to the left.

III. Analysis and Findings

In this section, we describe our approach for analyzing the collected information about the reported bugs and the findings from the analyses in terms of answers to the research questions formulated before in Section I.

A. Time Required to Triage and Fix Bugs

The stages in the life of a bug starts from its initiation at the point of identification to the date when the bug report is closed. Figure 4 summarizes these stages in the life of a bug in case of the Chromium project.

In accordance with the the Chromium "Bug Life Cycle and Reporting Guidelines" [1], the 'Fixed' and 'Verified' stages are described in Section II-A. The remaining stages are described as follows:

- *Unconfirmed:* "The default for public bugs. Waiting for someone to validate, reproduce, or otherwise confirm that this is a bug."
- *Untriaged:* "A confirmed bug that has not been reviewed for priority or assignment. This is the default for project members' new bugs."
- *Available:* "Confirmed and triaged, but not assigned. Feel free to take these bugs!"
- *Assigned:* "In someone's work queue."
- *Started:* "Actively being worked on."

Typically, there is lag in the movement of bugs between stages and substantial time also elapses when a bug remain in a stage while it is worked on by one or more developers.

Metrics: For the purpose of quantitative analysis, we define the following metrics:

- *Time-to-Fix:* is the time elapsed during a bug's transition from the beginning of 'assigned' stage to the beginning of 'fixed' state.
- *Triage Time:* is the time elapsed during a bug's transition from 'untriaged' stage to 'available' stage. During this period, a bug is reviewed for prioritization and possible assignment to developers who would be responsible to fix this bug.
- *Reopen Count:* is the number of times a bug is reopened.

In measuring the time-to-fix and triage time, we measure both the total time required to fix a bug and the average time of each fix attempt in case the bugs closed prematurely are reopened.

1) Time to Fix: In Table III, we present the total and average time (time-to-fix) in hours taken by the three cate-

Fig. 4. Different Stages of a Typical Chromium Bug's Life

TABLE III
AVERAGE TIME TAKEN TO FIX DIFFERENT TYPES OF BUGS

Chromium Bugs		Time (hours) Taken to be Fixed			
Type	**Count**	**Total**	**Average**	**Median**	**St. Dev.**
Security	1,934	2742642	1418	303.0	2869.80
Performance	2,410	3063376	1271	259.5	2549.18
Other Bugs	5,5211	87629837	1587	309.0	2957.72

gories (i.e., security, performance, and other) of bugs we have studied. As can be noticed in Table III, the performance bugs are found to have taken the shorter time on average to be fixed compared to the security bugs. The median time-to-fix for the performance bugs is also lower than that for the security bugs. The bugs other than security and privacy are found to have taken the longest time to be fixed.

TABLE IV
AVERAGE TIME-TO-FIX FOR BUGS FIXED IN 30 OR FEWER DAYS

Chromium Bugs		Time (hours) Taken to be Fixed		
Type	**Count**	**Total**	**Average**	**St. Dev.**
Security	1285	270967	210	188.67
Performance	1671	332438	198	185.90
Other Bugs	34732	6834552	196	192.76

2) Time to Fix in 30-Days Window: The high standard deviations (in the right-most column in Table III) encourages us to examine those bugs over a shorter 30-days window from the date the bugs were reported (i.e. bug reports opened) and entered into the 'unconfirmed' stage. Within this 30-days window, the total number of bugs fixed as well as the average time-to-fix for each categories of bugs are presented in Table IV. As noticed, the results in Table IV remains partly consistent with the results in Table III. That is, within the 30-days window, the performance bugs are still found to have been fixed faster than the security bugs, but the other bugs are found to have taken the shortest time to be fixed.

3) Triage Time: To have a better understanding of where much of the time is spent in the bug-fixing process, we examine the triage time spent for the three categories of bugs.

TABLE V
AVERAGE TRIAGE TIME FOR FIXING DIFFERENT TYPES OF BUGS

Chromium Bugs		Time (hours) Taken to be Fixed		
Type	**Count**	**Total**	**Average**	**St. Dev.**
Security	1934	2735112	1414	2888.43
Performance	2410	2957948	1227	2484.62
Other Bugs	55211	85315819	1545	2925.44

Table V presents the time in hours required on average for triaging each categories of bugs. As seen in the table, on average, the performance bugs are found to have taken the least time in triage followed by the security bugs.

TABLE VI
AVERAGE TRIAGE TIME FOR BUGS FIXED IN 30 OR FEWER DAYS

Chromium Bugs		Time (hours) Taken to be Fixed		
Type	**Count**	**Total**	**Average**	**St. Dev.**
Security	1285	307500	239	494.47
Performance	1671	324310	194	188.39
Other Bugs	34732	6841613	196	273.07

Again, driven by the high standard deviation (in the right-most column in Table V), we further investigate the triage time for only those bugs, which are fixed in a 30-days period from their opening. The average triage time taken by these bugs are present in Table VI. These results are in sync with the time-to-fix results in Table IV for bugs fixed in 30-days window. For the bugs fixed in 30-days period, the performance bugs are found to have been triaged faster than the security bugs, but the other bugs are found to have taken the shortest time in triage.

4) Reopen Count: It is found in the Chromium bug repository that sometimes closed bugs are reopened later typically when the associated bug resurfaces. For such bugs, our time-to-fix and triage-time metrics include the average of the periods between multiple openings and closures. Thus, results about time-to-fix and triage time presented before might have been affected by reopened bugs. We therefore, compute how many of each categories of bugs are reopened (i.e., opened more than once).

TABLE VII
NUMBER OF TIMES BUGS ARE REOPENED

Chromium Bugs		Reopened	
Type	**Count**	**Total**	**Percentage**
Security	1934	21	1.09%
Performance	2410	30	1.24%
Other Bugs	55211	623	1.13%

In Table VII, we present the number of reopened bugs of each category. As expected, for all categories, only a small percentage (i.e., below 2%) of bugs are reopened. The percentage of reopened security bugs is slightly higher than the that of the reopened performance bugs.

TABLE VIII
NUMBER OF TIMES BUGS REOPENED IN 30 OR FEWER DAYS

Chromium Bugs		Reopened	
Type	**Count**	**Total**	**Percentage**
Security	1285	10	0.78%
Performance	1671	26	1.56%
Other Bugs	34732	325	0.94%

Table VIII presents the number and percentage of bugs that are found to have (closed and) reopened within the 30-

days period from their first appearance. Within this 30-days window, the percentage of reopened performance bugs is found the highest, while the percentage of the reopened security bugs is found the lowest.

Indeed, the differences in the percentages of reopened bugs of different categories (as presented in Table VII and in Table VIII) are very small (i.e., less than 1%) and thus can be considered not significant. We now answer the first research question (i.e., RQ1) as follows:

Ans. to RQ1: *Overall, performance bugs are triaged and fixed faster than security bugs in the Chromium project.*

A probable explanation to this could be that the performance bugs are fixed faster with higher priorities because such bugs directly and frequently affect the end-users' experience, which a software vendor cares with immense importance.

B. Developers' Experience in Fixing the Bugs

Every bug, to be fixed correctly and efficiently, needs to be assigned to the developers with correct expertise. It is among the most important parts of bug fixing process. Each type of bugs has their own specifications and knowledge requirement for accurate execution for resolving the defects. Performance bugs require the depth of knowledge of the software system, compilation tool chain, execution bottleneck, and memory management in the least. Security bugs require a thorough understanding of possible security loopholes in the source code, in the used resources (e.g., third-party libraries) and in the operating system (OS) platform, on which the system under development is expected to function.

Metrics: Similar to the studies of Imseis et al. [7] and Zaman et al. [15], to estimate a developer's bug-fixing experience in terms of the number of bugs previously fixed by the developer. Based on this, for the bugs in each category, we compute the average developer *experience*.

TABLE IX
DEVELOPERS' EXPERIENCE (# OF PREVIOUSLY FIXED BUGS)

Bug Type	# of Distinct Developers	Cumulative # of Previously Fixed	Average Expertise
Security	108	1728	16.00
Performance	169	2259	13.37
Other Bugs	1488	9216	6.19

1) Results: In the rightmost column of Table IX, we present the average experience of the developers having fixed the three different categories of bugs. As seen in the table, the average experience of the developers who fixed the security bugs is the higher than that of the developers who fixed the performance bugs. Average experience of the developers who fixed the other bugs is the lowest by a substantial margin.

Again, we examine if the same trend sustains when we consider only those bugs, which are fixed in 30 or fewer days. The developers' average experience for these bugs fixed in this short time frame is presented in Table X. As can be noticed, the results in this table is still consistent with the results in Table IX. We, therefore, present the answer to the second research question (i.e., RQ2) as follows:

TABLE X
DEVELOPERS' EXPERTISE FOR BUGS FIXED IN 30 OR FEWER DAYS

Bug Type	# of Distinct Developers	Cumulative # of Previously Fixed	Average Expertise
Security	74	1385	18.72
Performance	145	2109	14.54
Other Bugs	1256	8707	6.93

Ans. to RQ2: *On average, more experienced developers are involved in fixing security bugs compared to the performance bugs or other bugs in the Chromium project.*

A likely explanation to this finding is that the security bugs are typically more complex for pinpointing the defect and difficult to test rather than other bugs. Moreover, the haste to close the security issues might also cause decline in due diligence required, which, in software engineering practice, often delay the process.

C. The Role of Operating System Platform

A software application working well on a particular platform often display buggy behavior when operated on another platform or environment. Which is why, modern software processes often include a 'staging' phase where the software system is executed, operated, and tested on its target operating system (OS) platform and environment.

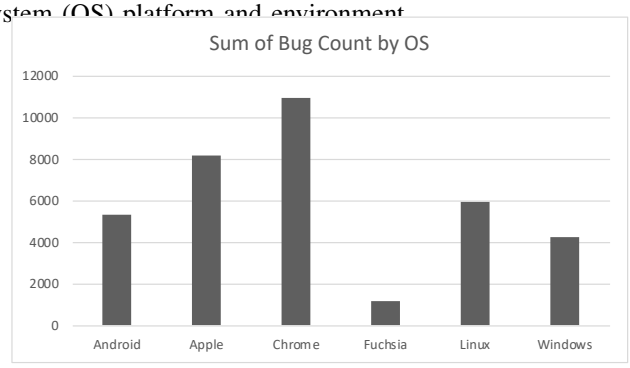

Fig. 5. Prevalence of Chromium Bugs on Different Operating Systems

Thus, we want to identify the OS platforms, on which the most bugs of the Chromium browser is exposed. Unfortunately, for a large portion (39.58%) of the reported bugs, the bug reports do not record the OS on which the bugs are triggered. For example, for the bug 1153449 in Figure 2, the OS information is not recorded. Hence, this part of our study focusing on the OS includes only those bugs for which the triggering OS platforms are recorded.

By parsing each bug report, we extract the information about which OS platform is affected by the corresponding bug. Figure 5 presents the number of Chromium bugs prevalent in different operating platforms. Each platform includes all bugs affecting all different hardware hosting the platform. For example, all bugs affecting the iOS, iPadOS, and MacOS are aggregated in Apple platform. Likewise, all bugs affecting

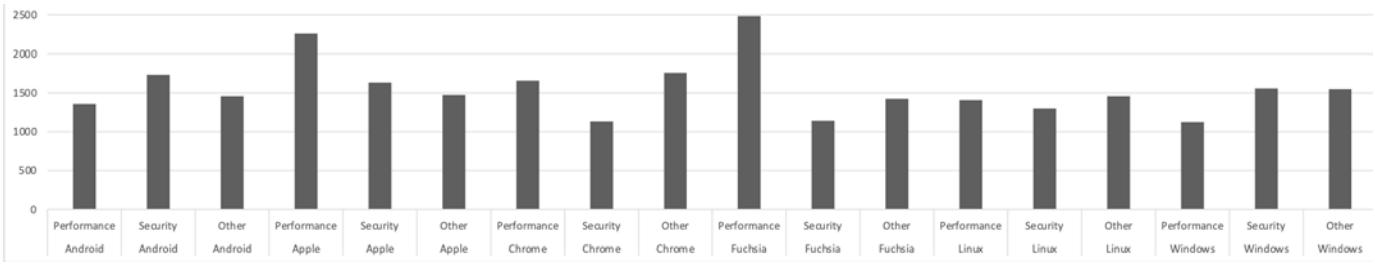

Fig. 6. Average Time (hours) Taken to Fix Different Categories of Bugs Triggered on Different Operating System Platforms

the Android operated devices (e.g., cell phones, tablets) are aggregated under the Android platform.

1) Results: For each OS, we compute how many of the bugs are triggered on the OS. Recall that, a particular bug can be triggered on more than one OS. For example, the bug 1153437 is triggered on Linux, Windows, and Apple (Mac) OS. As seen in Figure 5, Chromium bugs are the most prevalent on the Google Chrome OS, which means the Chrome OS platform triggers most of the bugs. Chromium bugs are the least prevalent on the Fuchsia OS possibly due to comparative smaller user base for this particular OS.

We further slice each of the three categories (i.e., security, performance, and other) of bugs based on the OS platforms on which they are triggered. Based on this additional OS-specific categorization, we plot (in Figure 6) the average time taken by each categories of bugs to be fixed. As seen in Figure 6, the performance bugs triggered on the Fuchsia and Apple platforms take the lengthiest time to be fixed, while the security bugs triggered on Fuchsia and Chrome OS are fixed in the shortest time compared to others. We now derive the answer to the third research question (RQ3) as follows:

Ans. to RQ3: *The bugs of the Chromium browser are the most triggered on the Chrome OS and the least triggered on the Fuchsia OS. Performance bugs on the Apple and Fuchsia platforms take the longest time to fix.*

IV. Threats to Validity

In this section, we discuss the limitations of our work, the threats to the validity of our findings, and our attempts to minimize those threats.

Construct Validity and Internal Validity: Our study includes bug reports opened over eight years period. Despite this large dataset, we might have missed important information as we did not include all bug reports in our study. This was not possible due to the sheer volume of Chromium bugs in the repository. Our bug classification method might have misclassified a few bugs. As mentioned before, not all the bug reports included all the information. Thus, important information might have been missed out, which might have affect our results. However, we could not find a way to mitigate this shortcoming.

External Validity: The findings from this study are derived from the study of bug reports for the open-source Chromium project. Thus, the results may not be generalizable to other browser projects such as the Firefox browser or to proprietary closed source projects such the Safari browser. Since our results are derived from an in-depth study of a large dataset spanning over eight years, we have high confidence in the generalizability of the results.

Reliability: The methodology of data collection, analysis, and results are well documented in this paper. The criteria including the date-range for bug report collection are also clearly mentioned in the paper. The Chromium bug repository is publicly available online [2]. Hence, it should be possible to replicate this study.

V. Related Work

There are many studies of different aspects of bugs [6], [9], [10], [11], [12], [13] and bug reports [16], [17]. Gegick et al. [4] specifically studied at security bug report by text mining. The work of Zaman et al [15] and Imseis et al. [7] are the most relevant to ours.

This work of ours is inspired by the work of Zaman et al. [15] and Imseis et al. [7]. Zaman et al. [15] studied the security bugs and performance bugs in the Firefox browser project. Recently, Imseis et al. [7] studied the same categories of bugs in the Chromium browser project. These two studies share some common research questions but they study two different browser projects. In our study, we follow the procedure of the work of Zaman et al. [15] as closely as possible but we study the Chromium project instead of Firefox.

Our study on the Chromium browser bugs is built on a dataset, which is different and larger than that of the work of Imseis et al. [7]. Thus, this work can be considered as a follow up to the studies of Imseis et al. [7] and Zaman et al. [15]. Zaman et al. [15] studied the performance bugs, and security bugs and other bugs in the Firefox browser project. They explored mainly three aspects, how fast bugs are fixed, who fixes the bugs, and other characteristics of bug fixes. Their study reported that on average security bugs were fixed 2.8 times faster than performance bugs and triaged 3.64 times faster. They also reported that security bugs were also reopened 2.5 and 4.5 times more than performance and other bugs. According to their study, security bugs were assigned to relatively more experienced developers for fixing.

Our study substantially differs from the work of Zaman et al. [15]. Although we closely follow their footsteps, we study

the bugs in the Chromium project. Instead of limiting our work to the bugs associated with the Chrome browser only, we included in our study the entire Chromium package. Similar to their finding, we also found the security bugs being assigned to more experienced developers. But, we found the performance bugs being triaged and fixed faster, which contradicts their result. While Zaman et al. [15] examined other characteristics of bug fixes, we included the OS-level aspect to our analysis, which their study does not have.

The recent work of Imseis et al. [7] is more similar to our study. Similar to ours, they also studied the three categories (i.e., performance, security, and other) of bugs in the Chromium project. Similar to the study of Zaman et al. [15], their work also examined how fast bugs are fixed, who fix the bugs, and other characteristics such as complexity. Similar to the work of Zaman et al. [15], their study also lacks the OS-level analysis, which our study includes.

While our study finds that performance bugs are fixed faster than security bugs, they reported the opposite. Similar to our finding, they also found that security bugs were assign to more experienced developers, where experience is estimated in terms of the number of previously fixed bugs by a developer. Note that, the study of Imseis et al. [7] is based on bug reports opened over only two months, "between March 2019 and April 2019" while ours includes a larger dataset consisting of bug reports opened over more than eight years (opened between *May 21, 2012* and *Nov 28, 2020*).

VI. Conclusion

Security and performance are two important software requirements especially more applicable to browser projects. We have studied the security and performance bugs in the Chromium browser project. Our dataset includes 189,668 closed Chromium bug-reports opened over more than eight years period. The findings are derived from an in-depth quantitative analysis of these large number of bug reports.

We have found that, on average, all categories bugs are almost equally reopened (and reassigned) once closed. There are variations in the prevalence of the Chromium bugs on different operating system platforms on which those bugs are triggered. We have also found that performance bugs are triaged and fixed faster than security bugs or other bugs in the Chromium project. Although more experienced developers are assigned to the security bugs, this category of bugs take longer to be triaged and fixed, possibly due to the higher complexity of the security issues. Some security bugs are found to have remained open for more than 2 years, which, at some level, is concerning, given that Chromium is a widely used platform.

The findings from this study advances our understanding of how security and performance bugs are prioritized and handled in practice. The insights drawn from this study are directly useful to the Chromium developers in shaping the strategy in dealing with bugs. In future, we plan to extend this work along two directions: (a) we plan to include even larger dataset across different projects for better generalizability of the results, (b) we want to include more qualitative analysis to draw insights into *why* the results appear the way they do.

References

[1] *Chromium Bug Life Cycle and Reporting Guidelines*. https://www.chromium.org/for-testers/bug-reporting-guidelines#TOC-Closed-bugs, verified: Feb 2022.

[2] *Chromium Bug Repository*. https://bugs.chromium.org/p/chromium/issues/list, verified: Feb 2022.

[3] E. Campos and M. Maia. Common bug-fix patterns: A large-scale observational study. In *Proceedings of the Empirical Software Enginerign and Measurement*, pages 404–413, 2017.

[4] M. Gegick, P. Rotella, and T. Xie. Identifying security bug reports via text mining: An industrial case study. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 11–20, 2010.

[5] Tricentis GmbH. *Software Fail Watch: 5th Edition*. https://www.tricentis.com/wp-content/uploads/2019/01/Software-Fails-Watch-5th-edition.pdf, verified: Feb 2022.

[6] M. Hamill and K. Goseva-Popstojanova. Common trends in software fault and failure data. *IEEE Transactions on Software Engineering*, 35(4):484–496, 2009.

[7] Joseph Imseis, Costain Nachuma, Shaikh Arifuzzaman, Minhaz Zibran, and Zakirul Alam Bhuiyan. On the assessment of security and performance bugs in chromium open-source project. In *Proceedings of the International Conference on Dependability in Sensor, Cloud, and Big Data Systems and Applications (DependSys)*, pages 145–157. Springer Singapore, 2019.

[8] Research Triangle Institute. The economic impacts of inadequate infrastructure of software testing. RTI Project Report 7007.011, National Institute of Standards and Technology, 2002.

[9] M. Islam and M. Zibran. A comparative study on vulnerabilities in categories of clones and non-cloned code. In *Proceedings of the 10th IEEE International Workshop on Software Clones*, pages 8–14, 2016.

[10] M. Islam and M. Zibran. On the characteristics of buggy code clones: A code quality perspective. In *Proceedings of the 12th IEEE International Workshop on Software Clones*, pages 23–29, 2018.

[11] M. Islam and M. Zibran. Sentiment analysis of software bug related commit messages. In *Proceedings of the 27th International Conference on Software Engineering and Data Engineering*, pages 3–8, 2018.

[12] M. Islam and M. Zibran. How bugs are fixed: Exposing bug-fix patterns with edits and nesting levels. In *Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing*, pages 1523–1531, 2020.

[13] M. Islam, M. Zibran, and A. Nagpal. Security vulnerabilities in categories of clones and non-cloned code: An empirical study. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 20–29, 2017.

[14] Md Rakibul Islam and Minhaz F. Zibran. What changes in where? an empirical study of bug-fixing change patterns. *ACM Applied Computing Review*, 20(4):18–34, 2021.

[15] Shahed Zaman, Bram Adams, and Ahmed E. Hassan. Security versus performance bugs: A case study on firefox. In *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR)*, pages 93–102. ACM, 2011.

[16] M. F. Zibran, F. Z. Eishita, and C. K. Roy. Useful, but usable? factors affecting the usability of apis. In *Proceedings of the 18th IEEE International Working Conference on Reverse Engineering*, pages 151–155, 2011.

[17] Minhaz F. Zibran. On the effectiveness of labeled latent dirichlet allocation in automatic bug-report categorization. In *Proceedings of the 38th International Conference on Software Engineering (ICSE) Companion*, pages 713–715. ACM, 2016.