An Investigation into Inconsistency of Software Vulnerability Severity across Data Sources

Roland Croft*†, M. Ali Babar*†, Li Li‡
* School of Computer Science, University of Adelaide, Adelaide, Australia, {roland.croft, ali.babar}@adelaide.edu.au
† Cyber Security Cooperative Research Centre, Australia
‡ Faculty of Information Technology, Monash University, Melbourne, Australia, {li.li}@monash.edu

***Abstract*—Software Vulnerability (SV) severity assessment is a vital task for informing SV remediation and triage. Ranking of SV severity scores is often used to advise prioritization of patching efforts. However, severity assessment is a difficult and subjective manual task that relies on expertise, knowledge, and standardized reporting schemes. Consequently, different data sources that perform independent analysis may provide conflicting severity rankings. Inconsistency across these data sources affects the reliability of severity assessment data, and can consequently impact SV prioritization and fixing. In this study, we investigate severity ranking inconsistencies over the SV reporting lifecycle. Our analysis helps characterize the nature of this problem, identify correlated factors, and determine the impacts of inconsistency on downstream tasks. Our findings observe that SV severity often lacks consideration or is underestimated during initial reporting, and such SVs consequently receive lower prioritization. We identify six potential attributes that are correlated to this misjudgment, and show that inconsistency in severity reporting schemes can severely degrade the performance of downstream severity prediction by up to 77%. Our findings help raise awareness of SV severity data inconsistencies and draw attention to this data quality problem. These insights can help developers better consider SV severity data sources, and improve the reliability of consequent SV prioritization. Furthermore, we encourage researchers to provide more attention to SV severity data selection.**

***Index Terms*—software vulnerability, severity assessment, data quality**

I. Introduction

Unpatched Software Vulnerabilities (SVs) can cause devastating consequences to organizations, and hence swift and effective remediation is essential [1]. Consequently, effective disclosure of SVs is a sensitive task. Identified SVs are often reported by multiple data sources, such as bug reports, vendor advisories, or vulnerability databases, to allow for timely and widespread dissemination [2].

With the sheer number of bugs and vulnerabilities that software developers encounter in modern software systems, developers also require intelligent and informed remediation plans [3]. Hence, SV severity assessment data is vital information provided by SV reporting sources that allows for better prioritization of fixing and patching efforts [4]. The severity of an SV is often influenced by exploitability and impact factors [5]; an SV that is both highly exploitable and has significant impacts should be considered critically severe. Severity rankings provide a natural ordering of SVs that can be used for initial prioritization.

However, as SV reporting data sources provide information at different stages of the reporting lifecycle, inferred independently by different analysts and scoring schemes, various reports may provide conflicting severity rankings for the same sets of SVs. Prior works have observed variations in severity scores for aggregated vulnerability databases [6], [7]. Whilst this inconsistency is expected due to the differences of each respective data source, it is an inherent problem. These independent data sources are used for the same tasks; SV assessment and prioritization. Hence, inconsistent severity rankings add confusion and unreliability. For instance, users may be unsure of which patches to prioritize if multiple data sources provide conflicting severity information. A set of vulnerabilities would be prioritized differently depending on the severity data source chosen.

Furthermore, researchers also rely on the reliability and correctness of SV data. Prior research efforts have used SV data to derive development insights [8], or to develop automatic assessment approaches [9]. Particularly, SV severity prediction has been a commonly explored task [5]. With conflicting information present in the available data sources however, the reliability and validity of such research is uncertain.

Whilst prior works have observed severity inconsistency [7], or investigated discrepancies amongst common scoring schemes [6], we are the first to analyse severity inconsistency within the full SV reporting lifecycle. Such insights are essential towards understanding severity information quality. We categorize three main SV reporting data sources [2]: 1) bug reports, which document initial identification; 2) vendor advisories, which perform initial disclosure; and 3) vulnerability databases that perform wider dissemination. We conduct our study through a large-scale investigation of the Mozilla Firefox development history, and its respective SV reporting sources: Bugzilla, the Mozilla Security Advisory, and the National Vulnerability Database (NVD). Our main contributions are:

- We provide an empirical investigation into the nature of inconsistency for severity rankings produced by different SV assessment data sources. Our analysis yields insights into the prevalence, characteristics, and correlated factors of such inconsistency. To the best of our knowledge, we are the first to examine such inconsistency across the full SV reporting lifecycle.
- We quantify the impacts that variations in these data sources can have on SV prioritization and prediction.

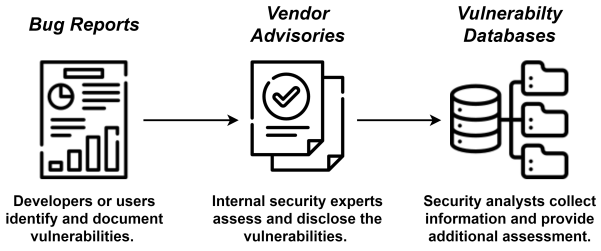

Fig. 1. An overview of SV reporting processes.

Primarily, we observe there to be weak correlation in the produced severity orderings for the three data sources that we have analyzed. This arouses a need for better standardization of the assessment information provided in these data sources, to increase reliability of prioritization schemes derived from such information. Furthermore, we observe that developer expertise is one of the key influential factors that leads to inconsistent severity rankings during initial reporting. We hence promote better education and training for this vital task. Finally, we also investigate the impacts that these issues can have for researchers, and observe that the choice of data source can heavily degrade the performance of severity prediction. We consequently advise caution and consideration for researchers of data quality when performing data selection.

The remainder of this paper is organized as follows. Section II presents the background knowledge and motivation for this study. Section III details our research methodology. We present our findings in Section IV, and discuss the implications of these findings in Section V. Section VI details the potential threats to validity of this research. Section VII describes related work. Finally, we conclude this paper in Section VIII. We have made our dataset and scripts publicly available as a reproduction package [10].

II. Background

A. Software Vulnerability Reporting Practices

As disclosure and dissemination of SVs is a sensitive task, both developers and security experts make efforts towards a thorough reporting lifecycle [2]. Figure 1 displays the standard SV disclosure process for large organisations, which contains three main stages: bug reports, vendor advisories, and vulnerability databases. Each stage provides independent severity assessment of SVs. We conducted our study on the Mozilla Firefox project, so we describe its respective data sources: Bugzilla[1], the Mozilla Security Advisory[2], and NVD[3]. The flow of information in Figure 1 adds time delays in reporting for each data source. We found that SVs were disclosed in the Mozilla Advisory a median of 65 days after Bugzilla. Disclosure in NVD takes an additional median of 13 days.

Bug Reports. Upon identification, SVs are reported to developers in the form of bug reports to a bug tracking system. Bug reports provide a description and basic assessment of identified bugs, to enable developers to implement a patch [11]. Unlike the latter two data sources, bug report severity is not necessarily specific to SVs; regular defects are also assessed. However, bug report severity assessment is still highly important for SVs, as it is used to inform prioritization [11].

Bugzilla is a popular bug tracking system that was originally developed and now used by the Mozilla project [12]. Bugzilla reports are usually assigned a severity score by the reporter of the SV. There are four classes of severity that may be assigned[4]; *S1-S4*. However, these ratings were only introduced in 2020, so we have manually mapped old Bugzilla severity classes to *S1-S4* based on their definition: *blocker (S1)*, the bug significantly impacts users or causes data loss; *critical* or *major (S2)*, the bug severely impairs functionality and a satisfactory workaround does not exist; *normal (S3)*, the bug blocks non-critical functionality and a workaround exists; and *minor (S4)*, the bug has low or no impact to users.

Vendor Advisories. SVs that are recorded in a bug tracking system are also independently disclosed to the public through vendor advisories that describe and document all SVs of a particular vendor or product. The Common Vulnerabilities and Exposures (CVE) system provides a reference system for SVs [13]. Vendor advisories often assign unique CVE IDs to SVs through a CVE Numbering Authority (CNA) [13].

The Mozilla Foundation maintains the Mozilla Foundation Security Advisory [14], and also acts as a CNA. This advisory is regularly updated, in correlation with new product releases. Severity ratings are independently assessed by the Mozilla Security team during the vulnerability remediation process [14], and are also added as a keyword to the associated Bugzilla report once determined. The severity ratings are estimated through the expected exploitability and user impacts, and fall under four classes[5]: *critical*, *high*, *moderate*, or *low*. For example, High severity SVs are "exploitable vulnerabilities which can lead to the widespread compromise of many users requiring no more than normal browsing actions."

Vulnerability Databases. Lastly, vulnerability databases aggregate information from a variety of vendor advisories and other sources, to provide a standardized collection of disclosed SVs. Unlike the prior two data sources that are maintained internally by an organization, vulnerability databases are maintained externally by a third-party, and contain information relating to a variety of vendors and products. Besides NVD, there are many independently maintained vulnerability databases that operate towards similar purposes, such as Exploit-DB[6], Snyk Vulnerability Database[7], or IBM X-Force[8].

However, vulnerability databases are not necessarily fully independent of each other as many aggregate information from a wide collection of resources, including other databases [6]. NVD is considered as the foremost vulnerability database

[1]https://bugzilla.mozilla.org/
[2]https://www.mozilla.org/en-US/security/advisories/
[3]https://nvd.nist.gov/
[4]https://wiki.mozilla.org/BMO/UserGuide/BugFields
[5]https://wiki.mozilla.org/Security_Severity_Ratings/Client
[6]https://www.exploit-db.com/
[7]https://snyk.io/vuln
[8]https://exchange.xforce.ibmcloud.com/

due to its thorough maintenance and integration with CVE. Consequently, other databases exhibit heavy overlap with NVD [6], so we only consider NVD for our analysis.

NVD is built to synchronize with the CVE list, and hence waits for CVE IDs to appear. NVD analysts then examine each SV and add enhanced information [15], such as severity, type, and affected versions. NVD assigns a severity score to SVs using the Common Vulnerability Scoring System (CVSS) [16]. There are currently two active CVSS versions: CVSS 2 and CVSS 3. CVSS 3 was introduced in 2015 to account for the criticized lack of granularity of CVSS 2 [16]. However, we considered CVSS 2 for this study, as data for CVSS 3 is not as complete due to its late introduction, and previous SVs are still relevant in modern contexts. For CVSS 2, a severity ranking is assigned to SVs based on a numeric score that ranges from 0-10: *Low* (0-3.9), *Medium* (4.0-6.9), and *High* (7.0-10.0). This score is calculated through a formula that assesses exploitability and impact metrics.

B. Motivating Example

As a motivating example, we consider the severity rankings and subsequent prioritization of a small subset of SVs. Table I displays the severity rankings for each assessment source for all vulnerabilities fixed in Firefox 77 and disclosed in the Mozilla Advisory entry MFSA2020-20. Whilst each assessment source uses individual severity rating schemes, all schemes follow an ordinal scale. Although we outlined up to four levels of severity in Section II-A, each data source only uses three different categories for this example. Hence, we have converted each scheme to their relative rankings, with 1 being the most severe and 3 being the least severe. Table II displays the natural ordering, and hence base prioritization scheme that would be inferred from these severity rankings.

We can see that even for this small subset of vulnerabilities there are large variations in the rankings and subsequent ordering. For instance, only two out of the ten SVs (V2 & V8) receive a common ranking across all three sources. Furthermore, V6 and V7 are prioritized first by the Mozilla Advisory, but last for NVD, despite both these data sources receiving assessment from security experts.

This inconsistency adds unreliability and confusion to the severity data, and makes us question the accuracy of prioritization schemes inferred from this information. For instance, what values should we trust and which ordering is optimal? Furthermore, we question the validity of research outcomes derived from such datasets, as they would inherently change depending on the selected data source. These questions motivate our analysis into the full extent of this issue; its characteristics, factors, and impacts.

III. Research Methodology

A. Research Questions

To analyse the characteristics, causes and impacts of inconsistency for SV severity rankings across data sources, we aim to address the following three Research Questions (RQs):

TABLE I
VULNERABILITY RANKINGS OF MOZILLA FIREFOX VULNERABILITIES THAT WERE FIXED IN RELEASE 77. 1 (RED) IS MOST SEVERE, 2 (ORANGE) IS MODERATELY SEVERE, AND 3 (YELLOW) IS LEAST SEVERE.

ID	Bugzilla ID	Severity Rankings		
		Bugzilla	Mozilla Advisory	NVD
V1	1619305	2	1	1
V2	1620972	1	1	1
V3	1623888	2	3	2
V4	1625333	2	1	1
V5	1629506	2	3	2
V6	1631576	3	1	3
V7	1631618	1	1	3
V8	1632717	1	1	1
V9	1637112	3	2	3
V10	1639590	2	1	1

TABLE II
THE PRIORITIZATION SCHEME FOR SVS FIXED IN FIREFOX 77 THAT IS INFERRED DIRECTLY FROM THE SEVERITY RANKINGS IN TABLE I.

Priority	Data Source		
	Bugzilla	Mozilla Advisory	NVD
1st	V2, V7, V8	V1, V2, V4, V6, V7, V8, V10	V1, V2, V4, V8, V10
2nd	V1, V3, V4 ,V5, V10	V9	V3, V5
3rd	V6, V9	V3, V5	V6, V7, V9

RQ1. What is the nature of inconsistency for severity reporting schemes? We first aim to illustrate and characterize the nature of this problem by detailing the inconsistency of SV severity data and reporting schemes for three different SV data sources of the Mozilla Firefox dataset. We seek to raise awareness of this issue, so that developers may better judge the reliability of SV prioritization, and so that researchers have more consideration of severity data quality and consistency.

RQ2. Which factors influence inconsistent rankings for SV severity assessment during early stages? Aside from inconsistent scoring schemes and experts, variations in severity rankings can also be introduced from inaccurate assessment [4]. Misjudgment is most likely to occur during early SV assessment, due to less consideration and available information [9]. By identifying potentially correlated factors of initial SV severity inconsistency, we can speculate causal factors of misjudgment. These insights can potentially assist developers or project managers to pinpoint attributes that can improve SV severity assignment, and hence prioritization.

RQ3. How does SV severity data source inconsistency affect downstream predictions? Other than impacts to SV prioritization, severity data inconsistency can also impact researchers who analyse severity data. One such task that has been regularly investigated is severity prediction [5]. Researchers can use different data sources depending on the selected stage for prediction, e.g., for bug reports [9] or SV databases [17]. Through this RQ, we display the impacts on prediction performance that dataset selection can have, and hence motivate researchers to properly consider this issue.

B. Data Collection

We selected the Mozilla Firefox project for analysis for the following reasons. Firstly, it is one of the largest open-source

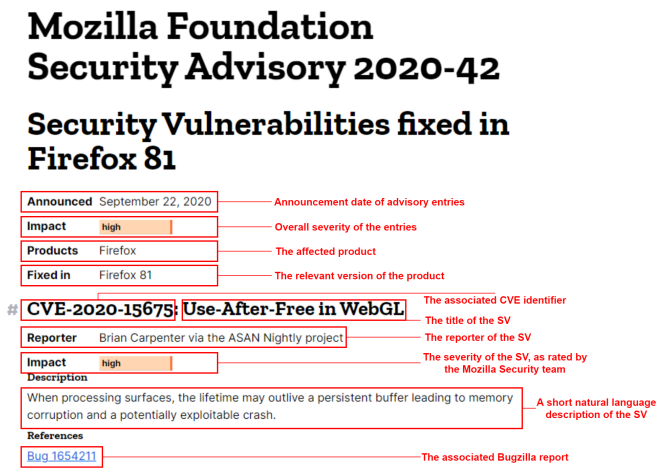

Fig. 2. An annotated example of the Mozilla Foundation Security Advisory.

TABLE III
NORMALIZATION OF DATA SOURCE SEVERITY RANKINGS.

Data Source	Severity Rankings			
	1st	2nd	3rd	4th
Bugzilla	blocker (S1)	critical/major (S2)	normal (S3)	minor (S4)
Mozilla Adv.	critical	high	moderate	low
NVD	critical	high	medium	low
Normalized	*critical*	*high*	*medium*	*low*

web browsers with over 200 million active users as of September 2021 [18]. Secondly, it has over a decade of active and publicly available development history. Thirdly, it maintains thorough and public SV reporting practices [14]. Finally, it has been the subject of many prior SV research studies [19], [20]. We collected SV data and severity information from each of the data sources described in Section II-A: Bugzilla, the Mozilla Advisory, and NVD.

To form our dataset, we first identified known SVs for Mozilla Firefox through the Mozilla Advisory [14]. We then scraped all the reported SVs from a 10 year time period of June 21st 2011 to June 1st 2021 (Firefox Release 5 to 89). Advisory data prior to this time period was not available. This provided us with a set of 1503 advisory entries.

Figure 2 displays an annotated example of an advisory entry. Each advisory entry provides a link to an associated Bugzilla report and CVE ID. We collected data for each individual bug report, and split advisory entries that had multiple associated bug reports. Additionally, we scraped the NVD details of each entry, as identified through the CVE ID. A CVE ID may refer to multiple bug reports. We removed duplicate bug reports or bug reports that were private at the time of data collection. This provided us with a final set of 2455 unique SV bug reports for 1329 unique CVE IDs.

To enable comparison of the severity rankings from each source, we normalized the labels of each source to a common format, as displayed in Table III. For NVD, we added a *Critical* severity class for CVSS 2 scores of 9.0-10.0. This allowed for better alignment of the other data sources, which had four classes each. We additionally added a *none* severity class for entries that were missing severity information.

We acknowledge that the severity classes of each source were not equivalent, due to differences in their definitions and scoring schemes. However, we considered their comparison valid as they used an ordinal ranking. We expect consistency in these rankings, as they are used to infer prioritization or other downstream tasks.

C. Statistical Modeling

To address RQ2, we aimed to identify factors that may lead to inconsistency in severity rankings for the initial Bugzilla severity assessment. As Bugzilla reports received the least security consideration, due to their fast communication, we speculate that inconsistencies to the Mozilla Advisory can be associated with potential misjudgment. We assume that correct Bugzilla severity assessment would induce severity rankings consistent with ones assigned and evaluated by experts.

Hence, to identify correlated factors for SV severity misjudgment, we used statistical tests and regression analysis, following practices established by prior works [21], [22]. Our dependent variable was whether Bugzilla severity was consistent with Mozilla Advisory severity. We did not consider NVD for this RQ as it is not as closely linked to Bugzilla reports as the Mozilla Advisory, and hence may introduce additional confounding factors. For explanatory variables, we scraped 18 attributes describing bug report metadata, as inspired by previous works [21], [22]. Table IV displays the 18 attributes and our rationale behind their inclusion for our task.

We first removed correlated attributes by using Spearman's rank-order correlation test [23] to determine highly correlated explanatory variables. We only retained one variable from each grouping that falls above a threshold value of 0.7, as this threshold value has been recommended in prior SE studies [21], [22]. Figure 3 displays the output of this correlation analysis; eight variables were removed. Attributes within the Fix Difficulty and Reporter Expertise categories were highly correlated to each other.

We then fitted a logistic regression model to the remaining 10 explanatory variables, as logistic regression models are simple but effective predictors for binary response variables [24]. We used the z-value of the regression coefficient to determine which variables had statistically significant coefficients, and were hence correlated with consistent SV severity assessment.

For RQ3, we aimed to investigate the impacts that data source can have on bug or SV severity prediction; a common problem explored by researchers in prior literature. Hence, we replicated standard practices by using textual description data as input [9], [17], to predict the normalized severity categories described in Table III. Following these practices, we preprocessed text descriptions through removal of stop words (using the NLTK and sklearn stopword list) and punctuation, conversion to lowercase, and stemming. The descriptions were then encoded using a bag-of-words model; we only extracted features for words that appeared in more than 0.1% of all descriptions [17]. We evaluated the same classifiers and tuned

TABLE IV
ATTRIBUTES OF A BUG REPORT THAT MAY INFLUENCE SV SEVERITY MISJUDGMENT.

Category	Name	Definition	Rationale
Fix Difficulty	Number of Patches	The number of patches that were submitted to fix the SV.	If an SV is difficult to remediate, developers may not be able to properly assess it due to a lack of comprehension.
	Number of Files	The number of files edited across all patches.	
	Number of Changes	The number of lines added, modified or deleted across all patches.	
Review Process	Fix Time	Time to fix a bug (resolved date - opened date).	Quickly resolved bugs may be ill-considered.
	Number of Comments	The number of comments on the bug report.	Extensive discussion may imply uncertainty of the evaluation of the SV.
	Number of Users	The number of users who comment on the bug report.	
	Number of CC	The number of users in the mailing list for the bug report.	Heavy interest may imply better SV assessment.
SV Nature	Description Length	The number of words in the description.	More explanation may provide easier assessment.
	Is Crash	Whether the string 'crash' in the description.	Crashes have been shown to be indicative of severity label noise [9].
Reporter Expertise	Reporter Bugs Filed	The number of bugs filed by the user.	Users who are good at reporting bugs may also be good at assessing them.
	Reporter Comments	The number of comments made by the user.	
	Reporter Patches Submitted	The number of patches submitted by the user.	Users who are good at resolving bugs may also be good at assessing them.
	Reporter Patches Reviewed	The number of patches reviewed by the user.	
	Reporter Bugs Resolved	The number of bugs resolved by the user.	
	Reporter Bugs Fixed	The number of bugs fixed by the user.	
	Reporter Bugs Verified	The number of bugs verified by the user.	
	Reporter Bugs Invalid	The number of bugs invalidated by the user.	
	Reporter Profile Age	The profile age of the user (current date - creation date).	Older users may be better at assessing SVs.

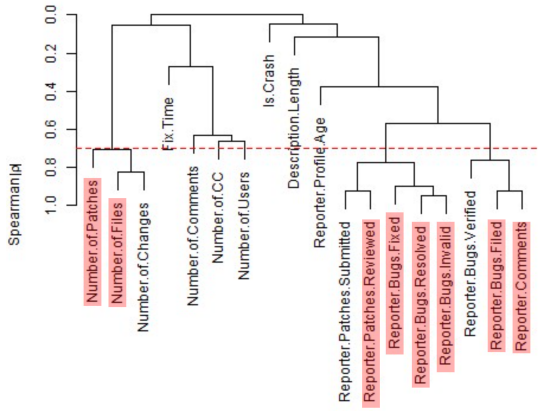

Fig. 3. Hierarchical clustering of variables according to Spearman's rank-order correlation. The dashed line indicates the high correlation threshold of 0.7. Removed variables are highlighted in red.

the same hyperparameters that were experimented with by prior works [17]. Full details of the experimental setup are available in our reproduction package [10].

Time-based validation methods have been shown to be important for this line of research [17]. Hence, we sorted our dataset by submission date of the report description, and then divided the dataset into a training, validation and test set through an 80:10:10 split. The validation set was used for tuning hyperparameters and selecting a model, and the test set was used for final evaluation of the optimal model. Model performance was measured using Matthew's Correlation Coefficient (MCC), as it has been found to be a less biased metric than other evaluation metrics [25]. The MCC score can range from -1 to 1, where 1 is the best value.

To thoroughly investigate the inconsistency of the SV severity labels across these data sources, we evaluated prediction performance for each SV description dataset when using each set of severity labels. For example, we built three prediction models for NVD descriptions when using NVD, Bugzilla and Mozilla Advisory labels separately.

IV. RESULTS

A. RQ1. What is the nature of inconsistency for severity reporting schemes?

Figure 4 displays the changes in severity rankings across the severity reporting schemes of the three data sources. Figure 5 presents a histogram of severity rankings for each data source.

Bugzilla severity rankings were more conservative; the majority of SVs were documented as Medium severity, and very few SVs were documented as Low or Critical severity. This limited diversity for Bugzilla reports may limit their practical usefulness, as most SVs would receive the same ordering under the Bugzilla severity scheme. Unlike Bugzilla, the Mozilla Security Advisory was skewed towards the most severe ranking. NVD exhibited the most uniform range of severity rankings, but still contained very few entries of Low severity. The flow of severity rankings displayed in Figure 4 reflected these changes in distribution.

SV severity is predominantly underestimated using the Bugzilla reporting scheme; 74% (1807/2455) and 48% (1169/2455) of Bugzilla reports were assigned severity rankings lower than that of the Mozilla Advisory and NVD, respectively. This implies that Bugzilla reports either pay little consideration to severity rankings, or lack the proper information and expertise to correctly assess SVs at reporting time. However, this incorrect assessment can lead to dire consequences, as this initial reporting information is used to establish prioritization. We found that over 94% (2317/2455) of bug reports were assigned a priority score using the initial information and severity data, before any security assessment from the Mozilla security team had occurred. Using a Mann-Whitney U test [26], we were able to favor the alternate hypothesis that Bugzilla SVs were often assigned a lower priority score when their severity rankings were inconsistent

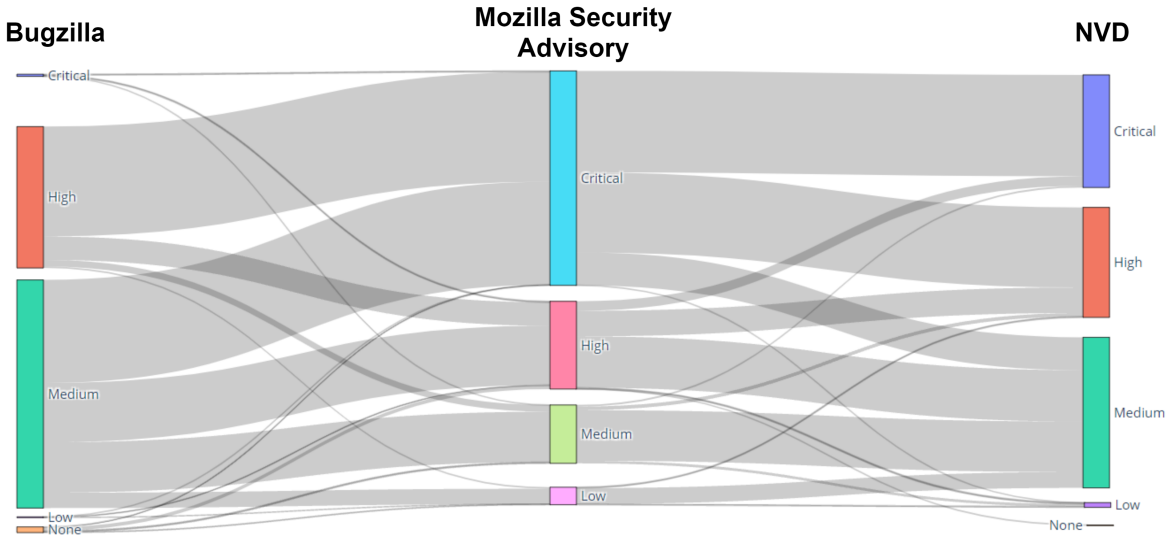

Fig. 4. Change in severity rankings across data sources.

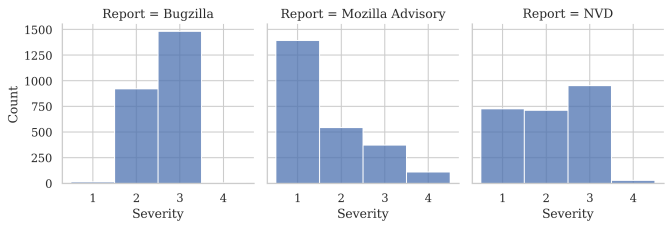

Fig. 5. Severity ranking distribution for each data source, where 1 = critical severity and 4 = low severity.

with the Mozilla Advisory ($U = 410981, p < 0.001$). This lower prioritization due to improper SV assessment can lead to delays in remediation.

Finding 1: *SV severity is often underestimated during first reporting. Misjudged SVs often receive lower priority.*

Furthermore, using Spearman's rank-order correlation coefficient [23], we observed Bugzilla severity rankings exhibited a weak correlation to the rankings of both the Mozilla Advisory ($\rho = 0.346, p < 0.001$) and NVD ($\rho = 0.260, p < 0.001$). This may be expected, as the Bugzilla severity scoring scheme is not exclusive to SVs, unlike the other two data sources. However, Bugzilla severity scores are still important as they are used to infer bug report prioritization. Hence, the lack of consistency between rankings for these data sources suggests that initial prioritization inferred from Bugzilla scoring schemes may be ill-suited for these security critical bugs.

There were also considerable differences in the severity rankings between the Mozilla Advisory and NVD, despite both of these sources receiving expert security analysis. Over 44% (1092/2455) of SVs received a higher severity ranking in the Mozilla Security Advisory than they did on NVD. Whilst it is unclear which severity ratings are more correct, this inconsistency at least implies that users ought to be aware of the nature and characteristics of each vulnerability database.

Finding 2: *Security experts and databases report SV severity differently.*

However, the Mozilla advisory and NVD severity rankings had a moderate to strong Spearman's rank-order correlation coefficient ($\rho = 0.624, p < 0.001$). Whilst their consistency was far from perfect, these two expert-based security sources at least had more correlation than the Bugzilla scoring scheme. This implies that expert assigned severity scores are more reliable than initial assessment scores.

We further observed that NVD severity rankings were not noise free either. Eight of the SVs in our dataset were referenced by two or more CVE IDs. This typically arose when SVs were announced in batches in the Mozilla Advisory, and were individually unspecified. Despite this duplication, the assigned severity was not always consistent. Five out of the eight (63%) duplicate NVD entries had different CVSS 2 scores. For instance, CVE-2016-1953 and CVE-2016-2805 were reported from a common bug report, but the former was assigned medium severity whereas the latter was assigned critical severity.

Finding 3: *Expert maintained vulnerability databases can still contain inconsistencies.*

B. RQ2. Which factors influence inconsistent rankings for SV severity assessment during early stages?

Aside from independent severity scoring schemes, misjudgment is another factor that can lead to inconsistency [4]. Inaccurate assessment or lack of consideration by users of a particular data source will add variation to severity rankings. For RQ2, we investigated potential causal factors that may lead to inconsistency between Bugzilla and Mozilla Advisory

TABLE V
STATISTICAL SIGNIFICANCE OF THE COEFFICIENTS OF BUG REPORT ATTRIBUTES FOR PREDICTING WHETHER THE SV SEVERITY RANKING WILL BE CORRECTLY ASSESSED.

Variable	z	$P > \|z\|$
Reporter Patches Submitted	-2.966	0.003
Fix Time	4.821	<0.001
Reporter Profile Age	-7.801	<0.001
Number of CC	-0.092	0.927
Description Length	-5.016	<0.001
Number of Comments	-1.088	0.277
Number of Users	0.599	0.549
Number of Changes	-0.769	0.442
Is Crash	-4.467	<0.001
Reporter Bugs Verified	-2.639	0.008

severity rankings, and hence potential misjudgment. We make the assumption that misjudgment is most likely to occur during early stages of the reporting lifecycle, as these stages receive the least consideration. Bugzilla severity scores are not as critically examined as the Mozilla Advisory ratings, which undergo thorough review for public disclosure.

We applied regression analysis to identify quantitative variables that were correlated to consistent severity assessment for Bugzilla reports. Table V displays the explanatory power for each of the attributes for predicting the likelihood that SV severity will be misjudged. We considered statistically significant coefficients as those with a p-value <0.01. Positively correlated significant coefficients are highlighted in green, whereas negatively correlated significant coefficients are highlighted in red. A positive correlation implies that an increase in the value of this attribute will increase the likelihood that SV severity is correctly assigned, whereas a negative correlation implies that an increase in this value will increase the likelihood that SV severity is misjudged.

We evaluated the goodness of fit of our logistic regression model by calculating the Area Under the Curve of the Receiver Operating Characteristic (AUC-ROC), as similarly used in prior works [21], [22]. Our model achieved an AUC-ROC value of 0.687. Although this is not a particularly strong value, we deemed it to be sufficient as regression coefficients are unaffected by goodness of fit [27].

Interestingly, the reporter expertise variables from Table IV were negatively correlated with correct severity assessment. Users with more experience, both through profile age and practical experience in submitting patches and verifying bugs, generally underestimated the severity when reporting an SV. This is unexpected, as one would often assume that users with more expertise would be better equipped to assess severity. Whilst we are unable to provide an explicit reason for this phenomenon, we can provide some speculation. Firstly, more experienced users may be busier or become less cautious, and hence have less time to spend filling reports. Furthermore, although these developers may be experienced, they still may not have the necessary security knowledge to accurately assess these unique defects, and triaging of security defects may be substantially different. Over 97% of the Bugzilla severity scores were assigned by the initial bug reporter.

TABLE VI
NUMBER AND PERCENTAGE OF CONSISTENTLY ASSESSED SVS FOR THE FIVE MOST DOMINANT CWE CATEGORIES (CASES > 30).

CWE ID	Category	# Cases	# Consistent	% Consistent
CWE 664	Improper Control of a Resource Through its Lifetime	1566	239	15.26
CWE 707	Improper Neutralization	188	91	48.4
CWE 264	Permissions, Privileges & Access Controls	123	44	35.77
CWE 693	Protection Mechanism Failure	34	14	41.18
CWE 399	Resource Management Errors	32	5	15.63

Finding 4: *Reporters with more experience are more likely to incorrectly assess SV severity.*

Bug reports with longer descriptions also had an increased likelihood of misjudgment. This is again unexpected as we would suppose that the more information that is available, then the more informed the SV severity assessment would be. However, upon inspection we found long Bugzilla descriptions to typically come from lengthy crash/error logs and stack traces. SVs relating to crashes were also generally underestimated. This is perhaps because they were so common; 880 of the 2455 (36%) SVs had associated crash data.

Finding 5: *Bug reports related to crashes or containing lengthy descriptions are more likely to be misjudged.*

Inversely, SVs that took a long time to fix were more likely to be assessed correctly. This may be because reporters who can recognize SVs that require more effort to fix also expend more effort towards assessing them.

Additionally, we examined whether SV type affects severity assessment. Different SV types exhibit different characteristics, hence one would expect that difficulty of assessment for each SV type also varies. The Common Weakness Enumerations (CWE) catalogs over 1000 different categories of SVs, using a hierarchical structure. We grouped CWEs to their highest level category to lower the dimensionality of our analysis, similar to Croft et al. [28]. If a CWE is contained by more than one category, we assigned it to the most frequent one. To increase the validity of our statistical analysis, we only considered high-level CWE categories that occurred more than 30 times in our dataset. This excluded 512 bug reports from the statistical test.

Using a Chi-Square test of independence [29], it was statistically significant that some CWE types were harder to assess than others ($X^2 = 153.74, p < 0.001$). Table VI displays the number and proportion of SV types consistently (and inconsistently) assessed for the most frequent CWE categories.

As discussed in RQ1, Bugzilla severity rankings were generally inconsistent with the severity rankings assessed by the Mozilla security team. From Table VI it can be seen

TABLE VII
SEVERITY PREDICTION MODEL PERFORMANCE FOR DIFFERENT SV DESCRIPTION AND SEVERITY RANKING DATA SOURCES. % CHANGE INDICATES THE PERCENTAGE CHANGE IN MCC FROM USING A UNIFORM DESCRIPTION AND SEVERITY DATA SOURCE.

Description Source	Severity Source	MCC	% Change
Bugzilla	Bugzilla	0.184	-
Bugzilla	Mozilla Advisory	0.042	-77.17
Bugzilla	NVD	0.124	-32.61
Mozilla Advisory	Mozilla Advisory	0.16	-
Mozilla Advisory	Bugzilla	0.051	-68.13
Mozilla Advisory	NVD	0.231	+44.38
NVD	NVD	0.217	-
NVD	Bugzilla	0.062	-71.43
NVD	Mozilla Advisory	0.086	-60.37

that resource related SV types (CWE 664 and CWE 399) were particularly difficult for reporters to correctly assess in Bugzilla reports; approximately 85% of severity rankings for these SVs were misjudged. Resource control and management can relate to a variety of weaknesses, such as crashes, race conditions, and memory leakage. These SVs may be difficult to assess due to their high frequency; reporters may not consider them as severe if they occur frequently. Furthermore, resource related SVs do not have as a distinct security context as other SVs, which may lead reporters to underestimate their severity. Improper Neutralization (CWE 707) and Access Control related SV types (CWE 264 and CWE 694) were more frequently consistent with the Mozilla Advisory severity rankings. This may be because the security impacts of these types are more well known [30] and hence easier to assess.

Finding 6: *SV type influences severity assessment accuracy. Resource related SVs are harder to assess.*

C. RQ3. How does SV severity data source inconsistency affect downstream predictions?

Table VII displays the performance of the tuned prediction models for each data source. The performance for all tasks was lower than the reported performance of prior works [9], [17]. We attribute this to our smaller dataset size in comparison to prior works, as otherwise our implemented techniques were largely similar. We were able to reproduce the performance reported by Le et al. [17] when using all documented NVD entries, not just for Mozilla Firefox.

We observe the performance to differ for different data sources. When using a uniform description and severity label source, the NVD data source produced the best performing models. This high performance is likely credited to the targeted analysis and standardized reporting of the NVD; both the descriptions and severity rankings follow consistent guidelines and are reviewed by security experts. Bugzilla data performed worse than NVD, likely as it was missing consistency. Descriptions and severity rankings were often unchecked and written by various reporters. One may expect Bugzilla to perform better, as the Bugzilla descriptions were usually longer than the other sources and hence may have provided more information. However, as observed in *Finding 5*, lengthy descriptions often contained noisy crash/error logs, which appear to be uninformative to the prediction model. The imbalanced distribution of severity ranking classes for Bugzilla data may also negatively influence the prediction performance. Finally, the Mozilla Security Advisory data performed worst, potentially because many advisory entries were missing meaningful descriptions. Entries that contained a batch of bug reports were not provided specific descriptions, e.g., the last two entries of MFSA2021-23[9]. The Mozilla Advisory appears to aim for simplicity, but this makes it a poor information source for data-driven tasks.

This creates an interesting decision for researchers, as each of these data sources has a different time gap for producing its description after the initial SV detection. Whilst NVD performs the best, it has the largest average time delay. Bugzilla produces descriptions instantaneously, but does not perform as well.

Finding 7: *Different data sources provide different performance for SV severity prediction.*

We also investigated transferring labels to descriptions of other data sources. We aim to investigate the extent of disparity between these sources. If any similarities exist between them, then knowledge should be able to be transferred from one source to another. For instance, can bug report descriptions also serve as a predictor of the severity labels assigned by security experts, or vice versa. However, the performance for the majority of these models noticeably dropped under this scenario; performance decreased by 77% in the worst case, which further highlights inconsistency in severity rankings.

The performance most noticeably degraded when using NVD descriptions with other severity label sources. If we consider NVD labels to be the most correct, as they are assessed at the latest stage and maintained by various security experts, then this performance decrease is expected as we are adding noise and inaccuracies to the labels by changing data source. The performance decreased by 71% when using Bugzilla labels in comparison to NVD labels. This again may be due to the heavy class imbalance of Bugzilla rankings. Conversely, model performance actually increased for the Mozilla Security Advisory descriptions when using NVD severity data, which attests to the quality of these labels. However, performance still decreased when Bugzilla reports were assigned NVD severity data. This may imply that Bugzilla descriptions have insufficient information to assess severity.

Finding 8: *Using inappropriate severity data sources (i.e. inconsistent labels) degrades prediction performance.*

V. IMPLICATIONS

A. For Developers and Managers

We first encourage SV reporters to exhibit more caution and security consideration when assessing SVs. Through *Finding*

[9]https://www.mozilla.org/en-US/security/advisories/mfsa2021-23/#CVE-2021-29967

1, we observed that developers often underestimated the severity of SVs during initial reporting in Bugzilla. Whilst this may be expected, due to the lack of dedicated and delayed analysis like the other two data sources, proper SV assessment is still vital at this stage. We found that 94% of bug reports were prioritized purely based on the Bugzilla severity ranking, and misjudgment often led to lower prioritization. By raising awareness of this issue, we hope our findings can motivate developers to spend more time and efforts on initial severity assessment. Managers may also choose to implement security specific assessment schemes for bug reports.

Accurate SV assessment can be difficult however. This is further observed through *Findings 4* and *6*, which suggest that inconsistencies in early stage assessment rankings are caused by a lack of developer expertise or knowledge. We promote the need for developers to be given adequate security training to perform this assessment. We found that assessment rankings were particularly inconsistent for Resource Management Errors (CWE-399) and Improper Control of a Resource Through its Lifetime (CWE-664). Hence, education should especially be provided for these types, so that developers can better recognize and assess them.

Furthermore, we suggest developers be wary of prioritization schemes directly inferred from base severity orderings. The inconsistency of severity rankings that we have observed in *Findings 1* and *2* implies that prioritization schemes that are solely dependent on this information are unreliable. If additional assessment efforts and resources are available, developers may find benefit from more intelligent prioritization schemes that also consider appropriate context, such as vulnerability conditions and exploit maturity, rather than just theoretical assessment. CVSS offers the ability for this extra assessment, through additional *temporal* and *environmental* metrics that can be assigned on top of the base score.

B. For Researchers

Firstly, researchers ought to continue to provide tool and knowledge support for SV severity assessment. Much research has been conducted into automated severity prediction [31], but this work is yet to make inroads into software development practices. Through *Findings 1, 4* and *6*, we observe that support is particularly needed for developers during early stage assessment of bug reports.

Without existing standardization, we similarly encourage researchers to investigate more varied severity data sources. Whilst much research has been done regarding severity prediction for NVD, very little work has been performed on SV severity prediction for bug reports or vendor advisories [5]. Existing bug report severity research only focuses on regular software defects [31]. However, in *Findings 2* and *6* we see that there is a heavy variation amongst these data sources. This may pose threats to external validity of existing research, as findings derived from one data source may not generalize to others. Hence, developers ought to consider more data sources due to their inconsistencies, or develop more robust methods that can handle such variation.

Whilst we encourage researchers to use more varied data sources, we also urge researchers to have more consideration for the impacts of their data selection. From *Findings 7* and *8*, we observed that the choice and consistency of data sources had significant impacts on severity prediction. Hence, researchers should be aware of noise and quality for their selected datasets.

Finally, we also suggest researchers to provide support for project specific severity assessment. We speculate that consideration of environmental and temporal metrics, such as those optionally available in CVSS, may provide more reliable and consistent assessment data. However, developers need support in acquiring this assessment, as current data sources only consider generalized base metrics.

C. For Security Experts

There is a need for standardization of SV severity assessment by security experts. A current lack of standardization results in different data sources producing different severity rankings and hence prioritization schemes, as seen in *Finding 2*. This inconsistency adds confusion and a lack of reliability to this data. Furthermore, a lack of standardized data sources negatively impacts the external validity of derived research outcomes, as previously discussed.

Whilst adoption and use of the Common Vulnerability Scoring Scheme (CVSS) [16] has made great strides towards standardized severity assessment, this issue is still far from solved. Independent analysis of SVs has been demonstrated to still lead to varying CVSS scores [6], [7], [32], due to potentially subjective and inaccurate assessment. Furthermore, use of the CVSS is also not commonplace yet; it was only utilized by one of our three data sources. Hence, there is a need for improved solutions towards standardization of severity assessment. Some solutions have been suggested for this problem, such as meta-scores [33] or majority voting schemes [7]. However, the validity and effectiveness of these solutions are currently un-evaluated.

Furthermore, we encourage security experts to make more efforts towards cleaning the records within these data sources, as this is another source of inconsistency. In *Finding 3* we observed some inconsistency within NVD from duplicate references to bug reports, and other researchers have observed similar noise issues [34]. It is the responsibility of security experts who maintain these data sources, to also ensure the quality of their documented data. Whilst security experts already make substantial efforts towards maintenance and accuracy, more thorough review and quality control of these data sources will potentially be a valuable effort.

VI. Threats to Validity

Construct Validity: A potential threat is our mechanism of comparison for SV severity across data sources. Due to the independence of these data sources, inconsistency is expected and may arise from a variety of factors. The ranking produced by one source would not be directly equivalent to that of another. Additionally, some manipulation of the categorization

schemes was required to allow for comparison. The use of CVSS 2 instead of CVSS 3 can also lead to some potential information loss from the assessment. However, we believe comparison to still be valid due to the ordinal scale of each scheme, and assert that inconsistency is an issue despite data source independence.

Internal Validity: We acknowledge that there may be confounding factors influencing correlation for RQ2. For instance, the reporter ID or the affected software product may also have high correlation to SV severity misjudgment. We will investigate these potential factors in future work.

External Validity: Like most empirical studies, our results may not sufficiently generalize to other applications or datasets. However, we have conducted our analysis on the Mozilla Firefox dataset, which is a large open-source application with many users. Furthermore, this dataset has been commonly analysed in prior research [19], [20].

Conclusion Validity: As we heavily relied on statistical analysis and hypothesis testing to infer our findings, our study may suffer from conclusion validity. However, we believe that the dataset we have used is sufficiently large to draw conclusions from. Furthermore, we have used a strict threshold of $p < 0.01$ for null hypothesis rejection.

VII. Related Work

A. Vulnerability Report Inconsistencies

Mining software repositories has become a popular area of empirical software engineering research [35]. However, the use of these data sources does not come without perils; the data is not necessarily clean and can exhibit significant noise [36], [37]. Our study contributes to this body of knowledge by highlighting a unique data quality issue that is present in SV reporting data sources; SV severity ranking inconsistency.

Tian et al. [9] identified that inconsistencies existed in bug severity data for duplicate bug reports, and hence demonstrated the unreliability of this data for bug severity prediction. Whilst duplicate bug reports can add inconsistency due to the differences from their respective reporters, we instead investigated the inconsistency of SV severity across data sources.

NVD itself has acknowledged that it may exhibit inconsistencies in severity scores to other data sources[10], due to subjective or inaccurate assessment. Several works have investigated inconsistency of CVSS scores for NVD in comparison to other vulnerability databases [6], [7], [32], but they concluded that vulnerability databases were relatively consistent, due to the large overlap in the sources that they collected their data from. Hence, we investigated severity ranking inconsistencies from independent, rather than overlapping data sources.

Finally, Dong et al. [2] investigated inconsistencies across NVD entries and their associated vulnerability reports, with a particular focus on vulnerability description and affected versions. They identified that information quality and consistency can be impacted as data is propagated from one source to another. We similarly investigated consistency of data propagation in the SV reporting lifecycle, but honed our focus on SV assessment and severity data.

[10]https://nvd.nist.gov/general/FAQ-Sections/CVE-FAQs

B. Automated Severity Prediction

Several approaches have been proposed by researchers to automatically assign a severity level to a bug or security report [31], to increase automation and reduce developer efforts. One of the first notable contributions was by Menzies and Marcus [38], who proposed a rule learning technique to automatically assess severity of bug reports. Gomes et al. [31] conducted a systematic mapping study of bug report severity prediction, and identified that the majority of works used learning-based approaches applied to unstructured text descriptions.

Similarly, researchers have developed automated approaches to assess the severity of an SVs. Le et al. [5] conducted a survey of automated software vulnerability assessment, and found that the majority of studies that aimed to predict severity used NVD text descriptions to predict CVSS scores. Recently, Le et al. [39] proposed to automatically assess severity directly from source code, to reduce reporting delay.

Hence, various data sources have been used by researchers to automatically assess severity at different stages; i.e., to increase automation for the initial reporting in bug reports, or for the more late and detailed analysis of NVD. To the best of our knowledge, we are the first to have compared and analysed inconsistencies in the data sources for this task.

VIII. Conclusion

In this study, we conducted a large scale analysis of SV severity reporting to identify characteristics, causes and impacts of inconsistency in SV severity data and reporting schemes, across the sources that it is reported. We performed this analysis using the Mozilla Firefox project as a case study. Predominantly, SVs were often ranked to be of lower severity at the time of reporting. This underestimation or lack of consideration for severity assessment caused such SVs to receive lower prioritization. We identified several potential causal factors for this initial inconsistent severity assessment, including that more experienced bug reporters often underestimated SV severity. We finally showed that this inconsistency in SV severity reporting schemes can severely degrade the performance of predictive tasks by up to 77%, depending on the information source used.

Through our findings we have drawn attention to this data problem and proposed several implications from this study. In summary, we recommend developers, researchers and security experts to deploy more consideration towards the quality and consistency of SV severity data sources. Furthermore, we have outlined some potential factors, and hence areas of improvement, that are correlated with correct assessment of SV severity at initial reporting.

Acknowledgment

This work has been supported by the Cyber Security Cooperative Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centre Programme.

REFERENCES

[1] H. Shahriar and M. Zulkernine, "Mitigating program security vulnerabilities: Approaches and challenges," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, pp. 1–46, 2012.

[2] Y. Dong, W. Guo, Y. Chen, X. Xing, Y. Zhang, and G. Wang, "Towards the detection of inconsistencies in public security vulnerability reports," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 869–885.

[3] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management—a systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, p. 106771, 2021.

[4] C. Fruhwirth and T. Mannisto, "Improving cvss-based vulnerability prioritization and response with context information," in *2009 3rd International symposium on empirical software engineering and measurement*. IEEE, 2009, pp. 535–544.

[5] T. H. Le, H. Chen, and M. A. Babar, "A survey on data-driven software vulnerability assessment and prioritization," *arXiv preprint arXiv:2107.08364*, 2021.

[6] P. Johnson, R. Lagerström, M. Ekstedt, and U. Franke, "Can the common vulnerability scoring system be trusted? a bayesian analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1002–1015, 2016.

[7] Y. Jiang and Y. Atif, "An approach to discover and assess vulnerability severity automatically in cyber-physical systems," in *13th International Conference on Security of Information and Networks*, 2020, pp. 1–8.

[8] Y. Wei, X. Sun, L. Bo, S. Cao, X. Xia, and B. Li, "A comprehensive study on security bug characteristics," *Journal of Software: Evolution and Process*, p. e2376, 2021.

[9] Y. Tian, N. Ali, D. Lo, and A. E. Hassan, "On the unreliability of bug severity data," *Empirical Software Engineering*, vol. 21, no. 6, pp. 2298–2323, 2016.

[10] R. Croft, A. Babar, and l. li, "Reproduction package for "an investigation into inconsistency of software vulnerability severity data"," 2022. [Online]. Available: https://figshare.com/articles/dataset/Reproduction_Package_for_An_investigation_into_inconsistency_of_software_vulnerability_severity_data_/16698124/2

[11] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 308–318.

[12] Mozilla, "Bugzilla." [Online]. Available: https://www.bugzilla.org/

[13] Mitre, "Common vulnerabilities and exposures." [Online]. Available: https://cve.mitre.org/

[14] Mozilla, "Mozilla security." [Online]. Available: https://www.mozilla.org/en-US/security/

[15] NIST, "National vulnerability database." [Online]. Available: https://nvd.nist.gov/

[16] FIRST, "Common vulnerability scoring system." [Online]. Available: https://www.first.org/cvss/

[17] T. H. M. Le, B. Sabir, and M. A. Babar, "Automated software vulnerability assessment with concept drift," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 371–382.

[18] Mozilla, "Firefox public data report." [Online]. Available: https://data.firefox.com/dashboard/user-activity

[19] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities," *IEEE transactions on software engineering*, vol. 37, no. 6, pp. 772–787, 2010.

[20] P. J. Morrison, R. Pandita, X. Xiao, R. Chillarege, and L. Williams, "Are vulnerabilities discovered and resolved like other defects?" *Empirical Software Engineering*, vol. 23, no. 3, pp. 1383–1421, 2018.

[21] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2146–2189, 2016.

[22] R. Paul, A. K. Turzo, and A. Bosu, "Why security defects go unnoticed during code reviews? a case-control study of the chromium os project," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1373–1385.

[23] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

[24] R. E. Wright, "Logistic regression." *Reading and understanding multivariate statistics*, pp. 217–244, 1995.

[25] J. Yao and M. Shepperd, "Assessing software defection prediction performance: Why using the matthews correlation coefficient matters," in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 120–129.

[26] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[27] R. B. D'Agostino, *Goodness-of-fit-techniques*. CRC press, 1986, vol. 68.

[28] R. Croft, D. Newlands, Z. Chen, and M. A. Babar, "An empirical study of rule-based and learning-based approaches for static application security testing," in *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2021, pp. 1–12.

[29] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[30] S. Barnum and G. McGraw, "Knowledge for software security," *IEEE Security & Privacy*, vol. 3, no. 2, pp. 74–78, 2005.

[31] L. A. F. Gomes, R. da Silva Torres, and M. L. Côrtes, "Bug report severity level prediction in open source software: A survey and research opportunities," *Information and software technology*, vol. 115, pp. 58–78, 2019.

[32] B. Schweigler, O. Nierstrasz, and P. Gadient, "An investigation into vulnerability databases," Master's thesis, University of Bern, Switzerland, 2020.

[33] VulDB, "Vuldb." [Online]. Available: https://vuldb.com/

[34] A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen, "Cleaning the nvd: Comprehensive quality assessment, improvements, and analyses," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2021, pp. 1–2.

[35] A. E. Hassan, "The road ahead for mining software repositories," in *2008 Frontiers of Software Maintenance*. IEEE, 2008, pp. 48–57.

[36] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 92–101.

[37] R. Croft, Y. Xie, and M. A. Babar, "Data preparation for software vulnerability prediction: A systematic literature review," *arXiv preprint arXiv:2109.05740*, 2021.

[38] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *2008 IEEE International Conference on Software Maintenance*. IEEE, 2008, pp. 346–355.

[39] T. H. Le, D. Hin, R. Croft, and M. A. Babar, "Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning," *arXiv preprint arXiv:2108.08041*, 2021.