Can We Trust the Default Vulnerabilities Severity?

Matteo Esposito[1], Sergio Moreschini[2], Valentina Lenarduzzi[3], David Hästbacka[2], Davide Falessi[1]
[1]University of Rome Tor Vergata — [2]Tampere University — [3] University of Oulu
m.esposito@ing.uniroma2.it; sergio.moreschini@tuni.fi; valentina.lenarduzzi@oulu.fi;
david.hastbacka@tuni.fi; falessi@ing.uniroma2.it

***Abstract*—As software systems become increasingly complex and interconnected, the risk of security debt has risen significantly, increasing cyber-attacks and data breaches. Vulnerability prioritization is a critical activity in software engineering as it helps identify and address security vulnerabilities in software systems promptly and effectively. With the increasing complexity of software systems and the growing number of potential threats, it is essential to have a systematic approach to vulnerability prioritization to ensure that the most critical vulnerabilities are addressed first. The present study aims to investigate the agreement between the default and the National Vulnerability Database (NVD) severity levels. We analyzed 1626 vulnerabilities encompassing 12 unique types of vulnerabilities associated with 125 Common Platform Enumeration identifiers belonging to 105 Apache projects. Our results show a scarce correlation between the default and NVD severity levels. Thus, the default severity of vulnerabilities is not trustworthy. Moreover, we discovered that, surprisingly, the same type of vulnerability has several NVD severity; therefore, no default prioritization can be accurate based only on the type of vulnerability. Future studies are needed to accurately estimate the priority of vulnerabilities by considering several aspects of vulnerabilities rather than only the type.**

***Index Terms*—Software Vulnerabilities, Security Debt, CWE, CVE, SonarQube, Empirical Software Engineering**

I. Introduction

Security debt has become a significant concern for software projects in recent years [1]. As software systems grow in complexity and connectivity, the risk of accumulating security-related issues, such as unpatched vulnerabilities, outdated software components, and insecure coding practices, has increased dramatically [2], [3]. Increasing cyber-warfare [4] and data breaches [5] threaten critical infrastructure and user privacy [6]–[8].

Security debt refers to accumulating security vulnerabilities and weaknesses that have not been addressed promptly. Like technical debt, security debt can grow over time and become increasingly difficult to manage [9]. Vulnerability prioritization determines which vulnerabilities pose the most significant risk to an organization and should be addressed first [10]. Organizations can focus their limited resources on addressing the most critical security issues and reducing their security debt by prioritizing vulnerabilities. Failure to prioritize vulnerabilities can lead to an ever-increasing security debt, making it more challenging to maintain an effective security posture and increasing the risk of a security breach [10]. Therefore, effective vulnerability prioritization is crucial to any comprehensive security strategy. Addressing this challenge requires software developers and security professionals to comprehend the factors contributing to security debt, develop effective strategies for mitigating it, and prioritize remediations.

The default severities assigned to rule violations may not be entirely accurate, primarily because most of the 12 rules have multiple severity levels associated with them. Furthermore, customization of priorities in each industrial context may be necessary since a one-size-fits-all approach may not prove effective. Therefore, the aim of this paper is to investigate the relationship between SonarQube (SQ) and National Vulnerability Database (NVD) severity [11], [12]. In particular, we focused on two aspects: (1) How many SQ vulnerability rules relate to a single Common Vulnerabilities and Exposures (CVE), and (2) the difference between the SQ rule default severity and the NVD severity levels.

To answer these questions, we conduct a data-driven analysis using a large dataset of CVEs and the official sonar rule database. Our results revealed that CVEs in the Apache Software Foundation (ASF) ecosystem are typically associated with a single rule violation (i.e., 92%).

Previous studies [13]–[21] focused on creating an analysis model for automatic vulnerability prioritisation and enhancing base metrics and data provided by security advisors or specialised tools. Although previous studies presented a range of primarily positive outcomes, to the best of our knowledge, no previous study comprehensively characterises the interplay between CVE, Common Weakness Enumeration (CWE), and SQ Rules. A notable knowledge gap exists in discerning the extent to which the latter effectively assist or potentially misdirect practitioners in prioritising vulnerabilities. Moreover, our investigation in the open-source software (OSS) domain amplifies its reach due to the OSS accessibility and availability. Hence, our main contributions are as follows:

- we provided the first characterisation of the relationship between CVE, CWE, and SQ rule to raise awareness regarding the ambiguity surrounding their linkage, and
- we performed a correlation analysis of NVD severity and default severity to determine whether the latter supports or misguides practitioners and researchers in their efforts to manage or study vulnerabilities.

Our findings provide valuable insights for software developers and security professionals in understanding the relationship between the two severity scores. These insights can inform the development of effective strategies for mitigating security debt in software projects, ultimately leading to more secure software products.

Paper Structure: Section II describes the related work, while Section III describes the study design. Section IV presents the obtained results and Section V discusses them. Section VI highlights some limitations and Section VII draws the conclusion.

II. Background & Related Work

Managing software vulnerabilities is essential for keeping users and data secure. The NVD uses CVEs to disclose a specific vulnerability in proprietary or open-source software publicly. CVEs standardise how security vulnerabilities are identified and tracked. CVEs aid organisations in managing and prioritising their effort [13], [14]. CVE specifies the vulnerabilities via CWE, a taxonomy that identifies and categorises common software weaknesses and vulnerabilities. Specialised tools can detect and prioritise vulnerabilities during software development by examining the source code without running it (i.e., static analysis).

SQ is an open-source platform that offers static code analysis to assist developers in identifying and resolving software vulnerabilities, bugs, and code smells[1]. We define the severity SQ assigns to each rule as "**default severity**". Vulnerability prioritisation is critical in software engineering as it helps identify and address security vulnerabilities promptly and effectively. Given the escalating complexity of software systems and the proliferation of potential threats, it is crucial to prioritize vulnerabilities based on their potential impact on the system [15], [16].

Many recent studies tackled the challenges posed by vulnerability prioritisation. Previous studies focused on creating an analysis model for automatic vulnerability prioritisation and enhancing base metrics and data provided by security advisors or specialised tools.

Huang *et al.* [17] formulate an analysis model using the fuzzy analytic hierarchy process. Their model aims to analyse the degree to which a specific vulnerability can affect the system, also considering human subjectivity in the final decision-making action in the prioritisation. Our work aims at unveiling the fundamental challenge of having a universal severity rating for each vulnerability. Hence multiple analysis models are required for each distinct analysis context.

Farris *et al.* [18] present a new software and network vulnerability management strategy called VULCON centred on time-to-vulnerability remediation and total vulnerability exposure. VULCON considers actual vulnerability scan reports, metadata about the identified vulnerabilities, asset criticality, available personnel, and custom performance metrics to prioritise vulnerabilities automatically. We address the issue of automatic vulnerability prioritisation as represented by SQ rules to test whether SQ rule default severity (i.e., an automatic approach) is a viable approximation of human expert prioritization (i.e., NVD scores).

Moreover, with the growing number of hardware and software vulnerabilities being discovered, manual classification of vulnerability types and prioritisation becomes increasingly tricky, justifying the need for automated machine learning classification [14]. Okutan *et al.* [19] developed an approach to curate vulnerability reports in real-time and map them to structured vulnerability attribute data using NLP and ML, automating the process and saving time compared to manual methods. Furthermore, Gonzalez *et al.* [20] highlights the importance of accurate and complete information in CVE reports to prevent software system vulnerability exploits. Hence, our work highlights the need for precise vulnerability prioritization to aid practitioners and researchers in accurately tackling the most relevant threat. Finally, Falessi *et al.* [21] showed that the negative consequences of a quality rule violation can change depending on the specific rule or context. Hence, in the same vein of our work, developers cannot effectively prioritise vulnerabilities unless they conduct context-specific validation and customisation of quality rules accordingly. Our work focuses on the correlation of the vulnerabilities among NVD and SQ Rules.

III. The Empirical Study Design

Our empirical study was designed as a case study following established guidelines [22]. In the upcoming sections, we detail the specific research questions and goals that drive our study and the procedures we used for data collection and analysis.

A. Goal and Research Questions

We formalized the goal of this study according to the Goal Question Metric (GQM) approach [23] as follows:

Investigate SQ Vulnerabilities and CVE, *for the purpose of* evaluation, *with respect to* their relation, *from the point of view of* developers, *in the context of* OSS.

Based on the aforementioned goal, we defined two main Research Questions (RQ_s) which serve as the primary focus of our investigation.

RQ_1. How many rules relate to a single CVE?

The SQ platform provides predefined rules that detect various types of security vulnerabilities in code. However, multiple CWE categories can impact a single CVE. On the other hand, multiple SQ rules can identify a single CWE. The multiplicity involved in this relationship can lead to ambiguity in understanding the root cause of the vulnerability and its remediation. Moreover, prioritizing which vulnerabilities to address can become tedious and challenging.

Therefore, investigating the relationship multiplicity can help us gather more accurate information and insights into the causes, remediation, and prioritization of vulnerabilities in software. Using CWEs as *trait d'union* between CVE and Sonar rule, we can grasp the specific SQ rules associated with a given CVE. Thus developers can focus their efforts on the most critical vulnerabilities and address them promptly and effectively. After establishing the relationship between CVEs and SQ rules and mitigating the ambiguity caused by the

[1]https://sonarsource.com/

many-to-many multiplicity, we can focus on prioritising the vulnerabilities. Hence we can ask:

RQ_2. Do default and NVD severity differ?

Vulnerability prioritization is a critical aspect of vulnerability management [24], as it enables developers to prioritize their efforts and address the most severe vulnerabilities first, reducing the risk of security breaches and potential harm to users. Analyzing the correlation between the default severity and the NVD base score (i.e., NVD severity) can help practitioners tailor vulnerability prioritization tasks to their specific needs and enhance our understanding of the vulnerability lifecycle. By understanding how the default severity and the NVD base score relate, practitioners can better evaluate the severity of vulnerabilities and prioritize their efforts accordingly. This approach allows practitioners to focus on critical vulnerabilities that pose a higher risk to their system's security rather than wasting time and resources on less significant vulnerabilities. Consequently, analyzing the correlation between these two factors can lead to a more effective vulnerability management process, ultimately enhancing the overall security posture of the system.

B. Context of the Study

The context of our study is the relationship between the default and NVD severity. We aim to identify the most common default severity associated with the NVD severity and their correlation. CVEs provide valuable information regarding multiple vulnerability metrics, including severity. More specifically, the NVD severity refers to the Common Vulnerability Scoring System (**CVSS**) [25] base score, a standard for assessing software systems' severity of security vulnerabilities. We focused on the CVSSv3, which provides a numerical and an ordinal score representing the severity of a vulnerability. Organizations can use those scores to prioritize their remediations to different security issues.

C. Project Selection

The project selection focused on the ASF ecosystem for two main factors: the first one is that we have access to a large number of CVEs associated with the ASF projects, enabling us to analyze a large amount of data. Furthermore, being open-source projects, source code is readily available to give us a broader view of the vulnerability impacting the specific project. Hence, we explored the relationship between NVD and default severity on a dataset containing 1626 vulnerabilities with 12 unique types of vulnerabilities associated with 125 common platform enumeration (**CPE**) identifiers belonging to 105 Apache projects.

D. Study Setup and Data Collection

This section outlines our data collection methodology. Our approach entailed accessing the CVE feed from the NVD encompassing vulnerability data spanning from 2004 to 2023.

To achieve our research goal, we filter out CVE items with null CVSSv3 fields (RQ_1) and without "apache" in the vendor part of the CPE. CPE is a standardized method for identifying IT systems and software products. the identifier consists of three components: a vendor name, a product name, and a version number, it is maintained by NVD and is an integral part of a CVE.

We merged the NVD base score with the matching default severity to conclude our analysis. This enabled us to investigate the correlation between vulnerability severity and default severity. It is noteworthy that we chose to conduct a correlation analysis [26] rather than an agreement analysis [27], [28], given that the variables under consideration are ordinal rather than nominal. To analyze the relationships between CVEs, CWE categories, and SQ rules (RQ_1), we extracted the relevant CVEs from the NVD feed and computed the multiplicity of the CVE-CWE relationships. We also extracted all SQ rules and computed the relationship between the rules and the CWE categories using a manually curated mapping. We decided to filter out all the CWEs with a one-to-many relationship with SQ rules and all CVEs with the same relationship to CWEs. Finally, we linked CVEs with their matching SQ rules via CWE (i.e., CVE→CWE→SQ Rules), obtaining the final dataset in which a single CVE is mapped to a single SQ rule. Hence, we can describe the relationship multiplicity among CVE, CWEs, and SQ rules in the ASF ecosystem.

E. Data Analysis

This section presents the data analysis procedure we employed in addressing our research questions.

To answer RQ_1, we linked CVEs, trough their CWE, to the matching SQ rules and obtain the final datasets that map one CVE to one SQ rule. Therefore, we compute the cardinality of CVE, CWE, and SQ rules relationships.

Finally, for RQ_2, we investigate the correlation between the default and the NVD base score severity computing Spearman's ρ. Hence, we gain crucial insights into the effectiveness of default severity in vulnerability management.

F. Replicability

We prepared a replication package[2] that includes all the scripts used in our analysis and the raw data. In addition, we have provided CSV files that contain the results for each research question. This enables other researchers to verify our findings and build upon our work.

IV. Results

A. RQ_1. How many rules relate to a single CVE?

Figure 1a presents how many CWEs are reported in a CVE; this table provides insight into the relationships between CVE and CWE entries and cardinality distribution across these entries. According to Figure 1a, 98% of 1,626 CVE entries (i.e., 1590 entries) report only one CWE. One CVE reports five CWEs. Figure 1b presents how many rules relate to how

[2]https://doi.org/10.5281/zenodo.8139908

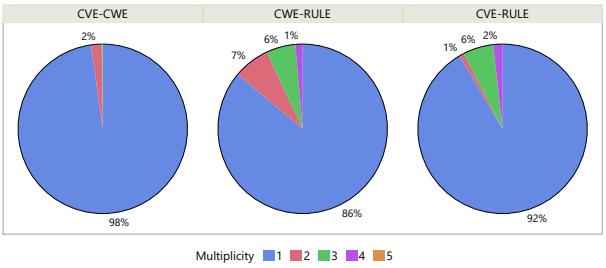

Fig. 1. Proportion of multiplicities among CVE, CWE and rules.

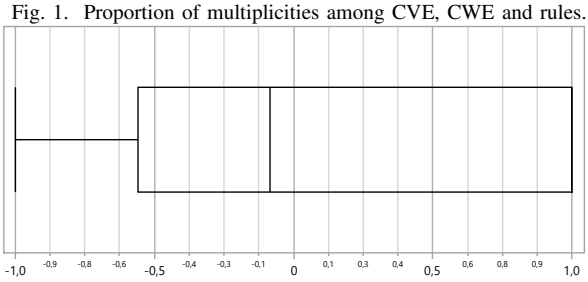

Fig. 2. Distribution of Spearman's ρ across projects.

many CWEs. According to Figure 1b, 86% of 72 CWE entries (i.e., 62 entries) relate to only one rule. Furthermore, five CWE entries relate to two rules, 4 CWE entries relate to three rules and one CWE entry relates to four rules.

Figure 1c presents how many CVEs relate to how many rules. According to Figure 1c, 92% of 342 CVE entries (i.e., 313 entries) relate to only one rule. Furthermore, three CVE entries relate to two rules, twenty CVE entries relate to three rules, and six CVE entries relate to four rules.

B. RQ_2. Do default and NVD severity differ?

Figure 2 presents Spearman's ρ correlation between default and NVD severity regarding project distribution. The mean ρ across projects resulted as 0.07; thus, the correlation is in average null. According to Figure 2, the distribution of ρ widely varies across projects. Specifically, most of the projects have negligible negative correlation and there is even a project where the correlation is fully negative. **This result suggests that we cannot trust the default severity level**.

One possible reason for the observed null correlation between default and NVD severity is that the same rule has several priorities in different CVEs. Figure 3 presents the proportions of NVD severity levels across 12 rules, i.e., all with at least two CVE entries. According to 3 the same rule has different NVD priorities in 11 out of 12 cases (i.e., 92%). **This result suggests that a default severity is not possible as it even changes for the same rules**.

V. Discussion

Regarding RQ_1, Figure 1c reveals that most CVE entries are tied to a single rule. Thus, the vulnerabilities identified in the ASF ecosystem are usually associated with a single rule. This finding suggests **that vulnerabilities in the studied ecosystem have a distinctive structure and can be detected using a single rule. Hence, identifying those vulnerabilities does not pose significant challenges**.

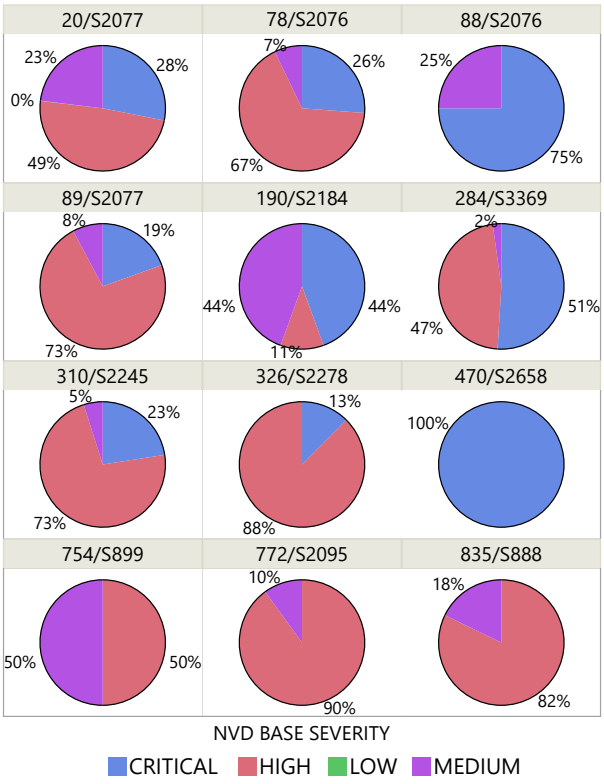

Fig. 3. Distribution of NVD Base Severity across CWEs/SQ rules.

TABLE I
SQ rule Type distribution

SQ rule Type	Count	Percentage
Bug	30	33%
Code Smell	20	22%
Vulnerability	42	45%

Regarding RQ_2, the low correlation presented in Figure 2 could suggest that a default severity is inaccurate. Therefore that is necessary to customize the priority in each industrial context. However, very interestingly, the fact that most of the rules (i.e., 92%) have several priorities, see Figure 2, suggests that context-based customization of rule priorities would be inaccurate. Specifically, Figure 2 suggests that **the priority of a rule violation must be established according to many factors other than the rule ID only**.

Finally, Table I presents the number of SQ rules with a particular type despite having "cwe" in their tags. This finding supports the conclusions of a previous study [29], which had identified inconsistencies between the type and the severity that SQ had assigned to Sonar-Issue affected classes.

VI. Threats to Validity

This section presents the threats to the validity of our study. We have categorized the threats by type, including Conclusion, Internal, Construct, and External validity [30].

A. Conclusion Validity

pearman's ρ is a non parametric measure of the strength and direction of a correlation between two variables, but due to multiple factors, this approach may hinder the conclusion's validity. For instance, the correlation coefficient is unreliable with a small sample size and cannot accurately represent the relationship between variables. Moreover, Spearman assumes a linear monotonic correlation between the variables; if the assumption were to prove invalid or not applicable, Spearman's correlation might not accurately measure the strength and direction of the association. We have mitigated this issue by ensuring a representative sample and checking the data distribution. We could mitigate this issue by using other non-parametric measures of correlation. Still, our analysis found that Spearman's ρ was more suitable than other correlation measures, such as Kendall's τ because it can effectively handle tied observations in the data. On the other hand, Kendall's τ relies on concordant and discordant pairs of observations.

B. Internal Validity

The internal validity of this study may be threatened by the lack of an official SQ rule - CWE mapping API or dataset. Nonetheless, we have taken measures to address this concern by conducting a comprehensive examination of the SQ documentation and meticulously linking each SQ rule to the corresponding CWE, as specified on Sonar's official website[3]. It is worth mentioning that the rule dump, containing the squid identifier (legacy name of SQ rules), pertains to SQ 7.5. Subsequent releases may have introduced additional rules and refined the existing ones. Therefore, future research should expand our rule set to mitigate the potential threats to internal validity.

C. Construct Validity

Construct validity concerns how our measurements reflect what we claim to measure [30]. Our specific design choices may impact our results, including our measurement process and data filtering. To face this threat, we based our choice on past studies and used well-established guidelines in designing our methodology [22], [23].

D. External Validity

External validity concerns how the research elements (subjects, artifacts, etc.) represent actual elements [30].

The present study utilized a substantial dataset obtained through scripted scraping of public records from the NVD concerning open-source ASF projects. Consequently, the study may be considered open-source focused. To address this limitation, future research should broaden the scope of the analysis to include industry projects. It is important to note that the NVD dataset was scraped during the initial week of June 2023, thus limiting the scope of replication efforts to that specific time frame. To facilitate the replication of our study, we have included a comprehensive replication package in Section III-F

[3]https://rules.sonarsource.com/java/

VII. Conclusion

In this section, we briefly draw our conclusions. Our study examined the relationship between the NVD severity and the severity assigned to violations of SQ rules. Our key finding was that relying on default severities for rule violations may lead to inaccuracies, as the same SQ rule may be associated with different NVD severities. This highlights the challenge of creating a universal severity rating for each vulnerability, as the vulnerability context and development team can have a significant influence. Therefore, customization of priorities in each industrial context is necessary [21]. In addition, we identified inconsistencies between the type and the severity that SQ had assigned to Sonar-Issue affected classes, which raises concerns about the reliability of SQ's classification system. Future studies are needed to accurately estimate the priority of vulnerabilities by considering several aspects of vulnerabilities rather than only the type. Specifically, the community should investigate how SQ rule severity can predict the likelihood of exploiting a vulnerability in the wild. Moreover, research should examine the impact of SQ on the vulnerability management process and mitigation. Finally, this research can benefit from, for instance, OSS project analysis to measure the impact of vulnerability management on real-world applications. Future works should focus on gathering OSS project analysis data, on a large scale, to further improve our impact in the field.

References

[1] K. Rindell, K. Bernsmed, and M. G. Jaatun, "Managing security in software: Or: How I learned to stop worrying and manage the security technical debt," in *International Conference on Availability, Reliability and Security, ARES 2019, 26-29*, 2019, 60:1–60:8.

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

[3] G. Tassey, *The economic impacts of inadequate infrastructure for software testing*, 2002.

[4] V. A. F. Almeida, D. Doneda, and J. de Souza Abreu, "Cyberwarfare and digital governance," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 68–71, 2017.

[5] S. Khan, I. Kabanov, Y. Hua, and S. E. Madnick, "A systematic analysis of the capital one data breach: Critical lessons learned," *ACM Trans. Priv. Secur.*, vol. 26, no. 1, 3:1–3:29, 2023.

[6] C. Ten, G. Manimaran, and C. Liu, "Cybersecurity for critical infrastructures: Attack and defense modeling," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 40, no. 4, pp. 853–865, 2010.

[7] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 973–993, 2014.

[8] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, 2021.

[9] C. Seaman and Y. Guo, "Chapter 2 - measuring and monitoring technical debt," in *Advances in Computers*, ser. Advances in Computers, M. V. Zelkowitz, Ed., vol. 82, Elsevier, 2011, pp. 25–46.

[10] B. Shreeve, C. Gralha, A. Rashid, J. Araújo, and M. Goulão, "Making sense of the unknown: How managers make cyber security decisions," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, May 2023.

[11] M. G. Siavvas, D. Tsoukalas, M. Jankovic, D. D. Kehagias, and D. Tzovaras, "Technical debt as an indicator of software security risk: A machine learning approach for software development enterprises," *Enterp. Inf. Syst.*, vol. 16, no. 5, 2022.

[12] M. Siavvas, D. Tsoukalas, M. Jankovic, *et al.*, "An empirical evaluation of the relationship between technical debt and software security," in *International Conference on Information society and technology (ICIST)*, vol. 2019, 2019.

[13] L. Bao, X. Xia, A. E. Hassan, and X. Yang, "V-SZZ: automatic identification of version ranges affected by CVE vulnerabilities," in *ICSE*, ACM, 2022, pp. 2352–2364.

[14] V. Yosifova, A. Tasheva, and R. Trifonov, "Predicting vulnerability type in common vulnerabilities and exposures (cve) database with machine learning classifiers," in *National Conference with International Participation (ELECTRONICA)*, 2021, pp. 1–6.

[15] Z. Zeng, Z. Yang, D. Huang, and C. Chung, "LICALITY - likelihood and criticality: Vulnerability risk prioritization through logical reasoning and deep learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1746–1760, 2022.

[16] J. Lin, H. Zhang, B. Adams, and A. E. Hassan, "Vulnerability management in linux distributions," *Empir. Softw. Eng.*, vol. 28, no. 2, p. 47, 2023.

[17] C.-C. Huang, F.-Y. Lin, F. Y.-S. Lin, and Y. S. Sun, "A novel approach to evaluate software vulnerability prioritization," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2822–2840, 2013.

[18] K. A. Farris, A. Shah, G. Cybenko, R. Ganesan, and S. Jajodia, "VULCON: A system for vulnerability prioritization, mitigation, and management," *ACM Trans. Priv. Secur.*, vol. 21, no. 4, 16:1–16:28, 2018.

[19] A. Okutan, P. Mell, M. Mirakhorli, *et al.*, "Empirical validation of automated vulnerability curation and characterization," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3241–3260, 2023.

[20] D. Gonzalez, H. Hastings, and M. Mirakhorli, "Automated characterization of software vulnerabilities," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 135–139.

[21] D. Falessi and A. Voegele, "Validating and prioritizing quality rules for managing technical debt: An industrial case study," in *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 41–48.

[22] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, 2009.

[23] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," *Encyclopedia of Software Engineering*, 1994.

[24] Y. Chang, P. Zavarsky, R. Ruhl, and D. Lindskog, "Trend analysis of the CVE for software vulnerability management," in *SocialCom/PASSAT*, IEEE Computer Society, 2011, pp. 1290–1293.

[25] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security & Privacy*, vol. 4, no. 6, pp. 85–89, 2006.

[26] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[27] B. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[28] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[29] V. Lenarduzzi, N. Saarimäki, and D. Taibi, "Some sonarqube issues have a significant but small effect on faults and changes. A large-scale empirical study," *Journal of Systems and Software*, vol. 170, p. 110 750, 2020.

[30] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell, *Experimentation in Software Engineering*. Springer, 2012.