

Between Risk, Recognition, and Necessity: How Open-Source Project Maintainers Perceive and Navigate CVEs Through Reporting and Resolving Vulnerabilities

Jessy Ayala
Department of Informatics
University of California, Irvine
Irvine, CA, USA

Steven Ngo
Department of Informatics
University of California, Irvine
Irvine, CA, USA

Joshua Garcia
Department of Informatics
University of California, Irvine
Irvine, CA, USA

Abstract

Researchers have investigated automated security impact and analysis techniques that leverage the content of Common Vulnerabilities and Exposures (CVEs). Understanding the perspectives of open-source software (OSS) project maintainers, especially those who historically lack a security background and have little to no targeted funding to secure OSS projects, is currently understudied. In this study, we investigate how OSS maintainers who have created or triaged security vulnerabilities perceive, use, and work around the CVE system. We use a mixed-methods approach through a listing survey ($n_1 = 77$), a Likert-scale survey ($n_2 = 109$), and interviews ($n_3 = 24$) to identify and categorize 36 factors into OSS maintainers' perceptions of CVE value, challenges associated with CVEs, and wants for future vulnerability identification and resolution features. We find that fix prioritization and assessing project health are the most important benefits for OSS maintainers, while producing fixes and how CVEs lack project context are the most challenging to overcome. Despite this, OSS maintainers express adaptability and openness to alternative systems, provided that they offer credibility, visibility, and ecosystem compatibility, and most desire less hands-on involvement while evaluating vulnerabilities. This work has implications for the future design of inclusive, OSS developer-centered vulnerability disclosure infrastructures, and gives a comprehensive insight into how current aspects of vulnerability identifiers work or do not work for the OSS community.

CCS Concepts

• **Security and privacy** → **Human and societal aspects of security and privacy**; **Vulnerability management**.

Keywords

Open-source security, Common Vulnerabilities and Exposures (CVE), Vulnerability management, Supply chain security

ACM Reference Format:

Jessy Ayala, Steven Ngo, and Joshua Garcia. 2026. Between Risk, Recognition, and Necessity: How Open-Source Project Maintainers Perceive and Navigate CVEs Through Reporting and Resolving Vulnerabilities. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security*



This work is licensed under a Creative Commons Attribution 4.0 International License. CCS '26, The Hague, Netherlands

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2871-6/2026/11
<https://doi.org/10.1145/3830454.3846649>

(CCS '26), November 15–19, 2026, The Hague, Netherlands. ACM, The Hague, NL, 15 pages. <https://doi.org/10.1145/3830454.3846649>

1 Introduction

The Common Vulnerabilities and Exposures (CVE) system is a core pillar of the cybersecurity industry and community, serving as an authoritative source for reported and known security vulnerabilities, with over 310,000 entries as of November 2025 [54]. Each time a vulnerability is discovered and reported to MITRE, which manages the CVE system, or an authorized CVE Numbering Authority (CNA), the vulnerability is assigned a CVE identifier (CVE-ID) [51], assuming that it is approved according to the CVE system's acceptance criteria [53]. CNAs, which include industry vendors, major open-source projects, bug bounty providers, and government agencies, are authorized to assign CVE-IDs for products and software packages within their scope, thereby supporting vulnerability management and remediation across the entire software supply chain. By coordinating these efforts, the CVE system enables interoperability between vulnerability databases and scanners throughout the software ecosystem.

Security practitioners across industry rely on the CVE system and the operations of the CNAs to identify and patch known vulnerabilities in their technology stack and dependencies, coordinate disclosure, share threat intelligence, and reduce risks of exploitation [9, 70]. Disruptions to the CVE system would degrade real-time threat coordination, leave automated scanners blind to untracked issues, and delay patching across the ecosystem [38, 64]. Yet the system's processes were largely designed around vendors and CNAs that operate with dedicated security staff and incentives to engage [6], which are conditions that do not hold for the open-source software (OSS) ecosystem, where vulnerability management is largely volunteer-driven and resource-constrained [13].

This mismatch makes OSS maintainers' experience with CVEs distinct from that of other CVE participants. Maintainers are frequently solo or small-team contributors with minimal to no security training [13], yet the CVE workflow expects them to advocate for identifiers, produce CVSS scores, and coordinate with CNAs as part of their volunteer efforts. While prior work has examined OSS maintainers' broader vulnerability management practices and their handling of bug bounty submissions [11, 13], neither characterized how maintainers engage with the CVE system specifically. We are the first to investigate the perspective of OSS maintainers who have had to create or triage CVE entries, and we provide the first ranking of the significance of CVE-related factors (e.g., usability challenges) from this perspective, offering insight into how the CVE system impacts and can better support OSS maintainers.

In this paper, we conduct a mixed-methods study that systematically identifies and quantifies the factors that affect OSS project maintainers' involvement in triaging CVE-assigned vulnerabilities. Unlike prior work, we ask the maintainers of the OSS project to rate the importance of factors, e.g., benefits and challenges, and elaborate on their rankings to allow researchers and platforms to prioritize exploring efforts to make the vulnerability disclosure and triaging process more accessible, i.e., for those with little or no security background. To do so, we explore the following:

RQ1: What are OSS maintainers' perceptions, benefits, and challenges when working with CVE-assigned vulnerabilities?

RQ2: How important are such CVE-centered benefits and challenges to OSS maintainers? Why are they important?

We conduct three studies, described in Section 3, to address our RQs: a survey ($n_1 = 77$) to list important perceptions, benefits, and challenges of CVE creation and triaging alongside wanted aspects during such processes (RQ1), a Likert-scale survey ($n_2 = 109$) to rank the importance of 36 coded factors (RQ2) from the listing study, and a semi-structured interview study ($n_3 = 24$) to contextualize overall survey results. In summary, OSS project maintainers cite fix prioritization and project health assessment as the most beneficial aspects of CVEs, producing fixes and severity scoring misalignments as the most challenging overall, and lacking project context and difficulties with CVSS scoring as the most significant usability challenges. Furthermore, we find that automated CVE creation, i.e., less OSS maintainer involvement for CVE advocacy and assignment, and risk assessment support are the most wanted capabilities when assigning and triaging CVEs.

2 Related work

CVE-related investigations. Several studies investigate the CVE system, particularly trends in vulnerability entries, the lifecycles of reported vulnerabilities, and inconsistencies and issues with vulnerability data. Neuhaus and Zimmermann [55] utilize topic models on the description texts of 39,393 CVEs to identify prevalent trends in available vulnerability data before 2010, and Chang et al. [17] study 22,521 CVEs across four years to better understand vulnerability trends within the CVE system and best inform security professionals. Lin et al. [44] examine CNA practices to compare trends between self-assigned and delegated vulnerabilities. Lim et al. [43] analyze the frequency of CVE records from 2017 to 2023 with a focus on ransomware vulnerabilities.

Yang-Smith & Abdellatif [88] examine the lifecycle of 3,362 CVEs in the Maven ecosystem to find that open-source maintainers prioritized patching severe vulnerabilities after disclosure, while Thomas et al. [78] conduct an in-depth analysis of 207 CVE entries for Siemens ICS devices to determine that the average time of the vulnerabilities existing before they were reported is 5.3 years. Wu et al. [87] also investigate the Maven ecosystem, examining CVE data to gain a deeper understanding of reachability and the complexities of reachable paths for reported vulnerabilities.

Dong et al. [23] compare 78,296 CVEs and 70,569 vulnerability reports to identify a high prevalence of inconsistent vulnerability software versions, which is also supported by a previous study by Nguyen et al. [56]. Guo et al. [33] examine over 133,000 CVEs and

found high rates of missing vulnerability aspects, including vulnerability type and root cause. Liu et al. [47] find, from a dataset of over 50,000 security bulletins and blogs from 36 CNAs, that industrial practices overlook the vulnerability aspects of attacker type, attack scenario, steps to reproduce, and vulnerability validation and exploitation. Zhang et al. [89] find 12,866 CVE entries that have inconsistencies in CVSS scores with other entries with similar vulnerability descriptions, and Sangaroonslip et al. [69] show that in more than 156,000 CVE entries, only 0.1% are privacy related. Li et al. [41] conduct a systematic study to better understand how research utilizes vulnerability databases, and Ruohonen et al. [67] examine the social network of the oss-security mailing list.

Collectively, this line of work establishes that CVE data is frequently inconsistent, incomplete, or delayed. CVE records alone, however, do not reveal who consults them or to what end. Such records cannot establish how these data deficiencies affect the practitioners expected to act on them, a gap our study addresses.

Vulnerability management, platform support, and severity scoring. The National Institute of Standards and Technology (NIST) provides authoritative guidance on creating internal vulnerability management programs [50, 71]. Previous studies provide insight into the vulnerability disclosure and management processes of existing groups, along with their issues and future directions [2, 10, 62, 65, 73]. Stevens et al. [73] discuss issues in Canada's vulnerability disclosure process and the need for more straightforward vulnerability disclosure and remediation paths in federal systems, and Piao et al. [62] examine the vulnerability disclosure policies of 264 AI vendors. Achuthan et al. [2] compare the disclosure process in FOSS to non-open source systems in India, while Perez et al. [65] compare the vulnerability management practices of IoT-centric vendors to other non-IoT-centric vendors by looking at more than 2,700 patches for IoT and non-IoT vulnerabilities. Other studies and reports also share personal experiences of navigating the vulnerability disclosure and management process [19, 52]. Moura & Heidemann [52] share their experiences of reporting a DNS vulnerability as researchers, and Chen et al. [19] documented their experience of reporting vulnerabilities for over 15,000 IoT backends, along with recommendations for improving disclosure practices.

Fischer et al. [24] conduct a time-series analysis on commits across over 50,000 GitHub repositories to investigate the effects of GitHub's platform security features, which are in support of vulnerability management for OSS maintainers. They found that all the features investigated had a positive effect on the security posture of OSS projects, but each feature differed significantly in its effect size. Ayala et al. [13] conducted in-depth interviews with OSS maintainers and identified particular barriers to their adoption of OSS platform security features, including a lack of awareness and the perception that such features are not necessary.

Prior work has explored methods to accurately determine CVSS scores and severity, especially in project-specific contexts, which remains an open challenge [49, 85]. Wu et al. [85] propose DIFFCVSS to support vulnerability prioritization within the Linux ecosystem by employing various techniques to assign different severity levels to reported vulnerabilities based on the target Linux version. Luo et al. [49] propose an automated vulnerability management system, CVECenter, which has reported success in managing over 8,000 CVEs in a specific Linux distribution and reducing the mean time

to fix bugs by providing an effective way to identify vulnerabilities that require severity re-evaluation. While these studies confirm that platform features improve security posture broadly, they do not isolate CVEs as a distinct coordination mechanism or rank which aspects of CVE engagement are impactful for maintainers.

Vulnerabilities and disclosure practices in OSS. There are two recent studies closely related to and similar to ours. Liu et al. [46] analyze the vulnerability disclosure practices of over 8,000 OSS projects and conducted a survey with 770 OSS maintainers on preferences in vulnerability disclosure models. Ayala et al. [13] conduct a mixed-methods study with OSS maintainers with previously vulnerable projects to gain a deeper understanding of their views on vulnerability management and platform security features.

Additional studies also focus on OSS projects and maintainers. Ayala et al. [11] conduct a mixed-methods study with OSS maintainers who interacted with *huntr*, a bug-bounty platform dedicated to OSS projects. Ramsauer et al. [66] demonstrate that established disclosure practices in OSS have an unintended side effect of opening up avenues of exploitability, e.g., fixes for vulnerabilities being discussed and then integrated privately, and Wermke et al. [82] further explored behind-the-scenes OSS vulnerability management practices. Pashchenko et al. [61] analyze OSS dependency vulnerabilities and compare them to the 200 most popular OSS Java libraries used by SAP, and find that most vulnerable dependencies are non-issues.

Tools and approaches have been proposed to support vulnerability tracing, management, and disclosure practices in OSS [40, 76, 86]. Tan et al. [76] propose PatchScout to help correlate OSS commits to CVEs in support of tracing patches, and Li et al. [40] propose PatchFinder to trace links to patches for OSS vulnerabilities using a framework with a patch retriever system and supervised fine-tuning. Wu et al. [86] conduct an empirical study to assess the quality of affected OSS libraries and their ecosystems for inconsistencies, then propose Holmes, an automated tool with a learning-to-rank technique, to perform such identifications.

Our findings differ from such prior OSS bug bounty and advisory studies, which gave little or no emphasis to fix prioritization, patching assistance, or missing project context, and identified producing fixes as, at most, a middling challenge. Our rankings instead show that maintainers value CVEs most for determining what to fix, while finding them least helpful for determining how to fix it.

Our work. As presented, there is an abundance of research that uses CVE data, e.g., individual CVE entries and CVE-IDs, as part of their training data and evaluation methods; however, such studies largely treat CVEs as static artifacts and do not focus on the quality of the CVE data or how practitioners interpret and utilize such data, which our study does from the perspective of OSS maintainers. Our study extends qualitative work that focuses on vulnerability management and practices in OSS by isolating CVEs as a distinct coordination mechanism within these processes and providing the first ranking of factor significance from those who have reviewed or advocated for CVEs, revealing unique perceptions and challenges in the research space. Our work also confirms the aforementioned challenges of using security features, but from the perspective of CVEs, and contextualizes how these challenges manifest within a standardized vulnerability identification infrastructure, and aims to understand how initiatives like the CVE program can better support

OSS vulnerability management. Our sample also differs in participant background from prior human-factors work on vulnerability management in operational settings [20], where nearly all of the participants were employed professionals and roughly half held at least partially managerial roles. Most of our participants maintain unfunded projects as volunteers, and most report no security background (i.e., 73% of listing survey participants, 65% of Likert survey participants, and 71% of interviewees).

3 Methodology

We leverage the methodology and three-study design of recent work that studied the perspectives of OSS bug bounty reviewers [11] to further examine the perspectives of OSS project maintainers, focusing on those who have triaged CVE-assigned vulnerabilities for their respective OSS projects. We conducted three studies to address our research questions: (1) a Listing survey to identify factors (RQ1), (2) a Likert-scale rating survey (RQ2), and (3) a follow-up interview study. The survey studies capture OSS maintainers' perceptions, challenges, and needs when working with CVEs, while the interview study provides deeper context on these responses. We include survey and interview instruments in our artifact [1].

The studies overlapped but followed a sequential deployment process. The Listing survey initially produced 33 factors. Interviews began in July 2025 after Listing participants indicated their willingness to participate and clarified the developing codebook, yielding three additional factors: limited scope, inconsistent formats, and more centralization. These factors were incorporated before the Likert-scale instrument was finalized in mid-August 2025.

Ethical considerations. To comply with GitHub's terms of service [31], we contacted maintainers with publicly available contact information advertised as reachable to the general public or via an external website, e.g., a personal homepage. We skipped recruiting OSS maintainers whose contact information was not immediately publicly available, e.g., avoiding using emails embedded in commits [79]. Our institution's ethics review board (IRB) approved all three studies prior to deployment. Participants signed consent forms that covered the study plans and their rights before data collection began; all of our studies are also GDPR-compliant. We include an extended ethics discussion as part of our artifact [1].

3.1 Listing survey study

To deduce factors, e.g., benefits, that impact OSS CVE management and triaging practices, we conducted an online survey between June 2025 and August 2025 with OSS maintainers from projects with validated CVEs from the GitHub Advisory Database, or from *huntr* or HackerOne bug bounty platforms ($n_1 = 77$).

Participant recruitment and piloting. We contacted OSS maintainers who own active projects with disclosed vulnerabilities from the *huntr* and HackerOne bug bounty platforms and had publicly advertised contact information in their GitHub security policy, e.g., `security@PROJECT.org`, or associated with the owning GitHub account. We also filtered GitHub security advisories to those with assigned CVEs, and were further deemed valid, i.e., contained a legitimate vulnerability. Two researchers then manually examined project profiles for metadata, discarding projects that do not fall under our IRB-approved protocol, which excludes subjects who reside

in OFAC-sanctioned countries and regions¹, e.g., *.ru or *.qq.com emails. Thus, we were left with 2,317 GitHub projects to potentially recruit from. To further comply with GitHub's terms of service [31], two researchers examined GitHub profiles, discarding those who do not have public contact information advertised to the general public, through an external website or social media. We reach out to 75% of such projects, 1,738 total, to reduce overlap between this study and our Likert study, and obtain a response rate of 5.9%, which is within the expected rate for online surveys (5–10% [84]).

We employ an open-ended listing approach for survey questions, called free-listing [14], which is commonly used in research when a domain is understudied, to gather a full breadth of responses. Although the limitations of modern vulnerability scanners and CVE systems have been studied in practice [6], we aim to gather insights from OSS project maintainers who have triaged such vulnerabilities, an underexplored perspective. We piloted the survey with 13 respondents to assess response quality and clarity of instructions for each listing question. We then improved question wording and encouraged respondents to list at least three factors per question across the categories shown in Table 2, e.g., *Overall benefits*.

Survey details. We asked participants to list the factors associated with CVE management and triaging in four categories: perceptions of the CVE process, benefits of the CVE system, overall and usability challenges when dealing with CVE-assigned vulnerabilities, and wanted features during the CVE process. We also asked participants to identify their OSS maintenance experience and industry background, as well as their security experience and project funding. We end with demographics to get further understand our population. **Data analysis.** We analyzed survey responses using exploratory open coding [68]. Two researchers worked through responses in batches, initially eight at a time and later six, independently assigning codes within each batch and then meeting to compare assignments, reconcile disagreements, and revise the codebook before moving to the next batch. Two coders obtained substantial agreement (Cohen's $\kappa = 0.78$) on the first 30 Listing responses and resolved disagreements through consensus, revisiting earlier responses whenever the codebook evolved. As each survey question elicited multiple listings per factor category, the codebook stabilized relatively quickly: 31.2% of responses yielded 26 of the 36 factors (72.2%) reported in Table 2. 36 responses were then coded in parallel with frequent consensus check-ins, resulting in 10 emerging codes. 11 responses were obtained after the final codebook was completed and were absorbed into existing codes, indicating saturation.

3.2 Likert scale survey study

The listing study revealed a detailed list of factors considered by OSS project maintainers, but not their relative importance. Thus, we conducted another study between August 2025 and November 2025, asking participants to rate the importance of each factor, e.g., *Overall benefits*, using a five-point Likert scale ($n_2 = 109$).

Participant recruitment and piloting. We emailed maintainers from 50% of the OSS projects described in Section 3.1, 1,159 projects total, 33% of which had overlap with the Listing survey study. We further expanded recruitment to include OSS projects with CVE-assigned vulnerabilities from OSV.dev [58], a comprehensive OSS

vulnerability database, yielding 471 more projects. Additionally, at most one-third of Likert survey participants were from the Listing survey, suggesting that most were newly recruited and helping to reduce potential sampling bias. Overall, we obtained 109 responses after contacting eligible participants, obtaining a 6.3% response rate (109/1,721), which is within the expected rate for online surveys (5–10% [84]) and higher than recent Log-Linear Bradley-Terry (LLBT) Likert-scale studies with maintainers (5.4% in [11]), with an overlap of 31.2% (34/109) between surveys.

Survey details. Survey participants were asked to rate each of the 36 factors on a five-point Likert scale (rows in Table 2). We ended with standard demographic questions to understand our sample.

Data analysis. Because our goal is to rank the factors against one another rather than report absolute agreement levels, we avoid summary statistics (e.g., means) as they treat Likert-scale responses as equidistant, which has been criticized by statisticians [21, 48, 83]. We instead fit a LLBT model to derive worth estimates (π) to capture the relative importance of factors on a preference scale [22]. Our LLBT-usage adapts the Paired Comparison Approach [21] and follows recent work [5, 13]. Participants rated all 36 factors on a five-point scale. We fit models per category because they capture distinct constructs. Per participant and category, we compared every rated pair of factors by the same participant: If factor f_j received a higher rating than factor f_k , then f_j was preferred to f_k ; equal ratings were treated as ties. This produced $X \cdot \frac{N(N-1)}{2}$ comparisons for X participants and N factors, e.g., benefits yield 3,052 comparisons. The pairwise preference probability between factors j, k follows directly from these estimates as $p(f_j > f_k | \pi_j, \pi_k) = \frac{\pi_j}{\pi_j + \pi_k}$.

3.3 Interview study

We asked consenting and interested Listing survey participants to participate in a remote semi-structured interview between July 2025 and October 2025 to learn more about why the factors they listed were especially important to them ($n_3 = 24$). Interviewee details and the interview guide are available in our artifact [1].

Participant recruitment and piloting. Interviewees were recruited from participants in the Listing survey study (Section 3.1). We invited Listing survey respondents who expressed interest to participate in remote semi-structured interviews if they had created or triaged at least two security reports with assigned CVEs, allowing us to capture a range of review experience levels. We piloted the interview protocol with four participants to evaluate the semi-structured format, and based on the resulting response quality, refined the interview questions to better focus our research objectives. In particular, we made minor updates to four interview questions to enhance their clarity and reduce the time spent on follow-up clarifications, and removed five questions entirely because they were repetitive. Revisions were confined to adding follow-up questions and making minor structural adjustments, with saturation reached and no further changes after the tenth interview. **Interview details.** We conducted semi-structured interviews with 24 OSS project maintainers, each lasting an average of 52 minutes. Prior to the interviews, participants received a study information sheet via email, and verbal consent was obtained at the start of each session. Participants were compensated with a \$20 virtual gift card. We began by asking participants about their project responsibilities, followed by why their listed benefits, challenges, and desired

¹OFAC sanctions are economic and trade embargoes issued by the Office of Foreign Assets Control (OFAC), an agency of the U.S. Department of the Treasury.

factors were particularly important. Due to time constraints, we did not cover all 36 factors shown in Table 2; instead, participants were encouraged to describe factors they considered most significant, which often led to discussion of other related factors. We then asked about their familiarity with the CVE Record Lifecycle [51] and experiences regarding CVEs for vulnerability triaging and assignment. Finally, we asked how they would consider automated techniques to improve the quality and speed of CVE evaluation.

Data analysis. All 24 audio-recorded interviews were transcribed and verified for accuracy by two researchers. We applied thematic analysis [15], starting with open coding [18] of each transcript and progressing to axial coding [18] to deduce recurring themes, e.g., 8 themes (Section 5.2). Throughout, we used memo writing and constant comparison [32] to surface and resolve discrepancies between within- and across-transcript instances and to develop higher-level thematic categories. Consistent with reflexive thematic analysis [15] and usable security approaches [5, 11, 82], we used iterative consensus coding rather than frozen-codebook inter-rater reliability (IRR). Two researchers independently coded the first 8 transcripts, then met over 3 calibration sessions. Both then independently coded the remaining 16 transcripts, reconvening per batch to resolve conflicts until no new codes emerged [18]. As our interview analysis followed an iterative consensus process with an evolving codebook, we do not report an IRR statistic. Instead, disagreements identified during coding were discussed and resolved, and earlier transcripts were revisited whenever new codes emerged.

3.4 Limitations

All of our data is self-reported, and participants' shared experiences may be incomplete or reconstructed after the fact. To mitigate this, we triangulate our RQs across the three aforementioned studies so that any factor surfacing in our results was either independently produced by participants in open-ended listing or corroborated across the Likert-scale and interview study. Each study was also piloted and revised in response to participant feedback, as described in the respective study subsections.

Participants may have varying experiences in triaging CVE-assigned vulnerabilities, some more than others, but we view those with less experience as still providing relevant perspectives or reflecting similar factors, e.g., *Overall challenges*. Further, since our study only recruits OSS maintainers who have prior experience creating or triaging at least two CVEs, it lacks perspective from OSS maintainers who have never engaged with the process. While our sample size, 24 interviewees and 186 survey respondents, is sufficient for the qualitative claims and ranked comparisons we make, it is not a probability sample of the broader OSS maintainer population. Table 1 shows the variation we captured in education, OSS experience, and security background. We also note that, because survey participation was uncompensated and OSS maintenance is largely volunteer work, our sample is likely to skew toward maintainers intrinsically motivated to discuss vulnerability disclosure.

4 Participants

Table 1 summarizes participants' (1) self-reported demographics and (2) experience. Our sample includes 77 participants from the listing study, 109 from the Likert-scale study, and 24 interviewees. Participants were primarily based in Europe and North America and

were predominantly male, consistent with prior qualitative studies of OSS developers, which report similar distributions [11, 12, 34]. Further, this reflects a sample largely composed of experienced OSS maintainers with substantial industry and project involvement.

Table 1: Participants across our three studies and demographics, various backgrounds, and project details. Acronyms listed represent, respectively: listing (L) study, interview (I) study, and Likert-scale (LS) study.

		L	I	LS
Gender	Man	67	20	93
	Woman	9	3	15
	Non-binary	1	1	1
Age	18-29	7	3	12
	30-39	32	8	57
	40-49	21	11	27
	50+	17	2	13
Residence	Australia	2	1	0
	Africa	2	0	8
	Europe	34	13	51
	Middle East	0	0	1
	North America	32	8	40
	South America	1	0	0
	South Asia	3	0	1
	Southeast Asia	2	1	2
	Other	1	1	6
Education	≤ Completed high school	2	0	11
	Trade/technical/vocational	2	1	2
	College, no degree	11	2	24
	Professional degree	3	1	5
	Bachelor's degree	28	9	21
	Graduate degree	31	11	46
Years working in industry	< 1 year	1	0	7
	1-5 years	5	1	10
	5-10 years	18	5	40
	10+ years	53	18	52
Years in OSS maintenance	< 1 year	1	0	3
	1-5 years	12	2	9
	5-10 years	25	10	42
	10+ years	39	12	55
Has a security background	Yes	21	7	38
	No	56	17	71
OSS project has sponsorship or public funding	Yes	29	9	32
	No	48	15	77
# of participants		77	24	109

5 Results

We identified 36 factors that OSS maintainers listed after having dealt with CVE-assigned vulnerabilities, falling under five categories regarding CVEs: perceptions, overall benefits, overall challenges, usability challenges, and maintainer wants. A full list is

Table 2: 36 factors identified from the listing survey study and interview study, divided into five categories as a result of software vulnerability management efforts regarding CVEs. Acronyms listed represent, respectively: existing related work (RW), the number of listing (L) survey study mentions, and worth estimates (π), for ranking responses, from the Likert-scale survey study. * indicates a tie between rankings in the Likert-scale survey study, i.e., worth estimates are equal. $-\dagger$ are factors that emerged exclusively from the interview study prior to deploying the Likert-scale survey study. An extended version of this table reporting confidence intervals for each corresponding worth estimate is included in our artifact [1].

	Factors	Description	RW	L	π
Perception	Critical to cybersecurity	CVEs are essential for timely & effective management.	[42, 63, 74]	21	0.323
	Should be reproducible	CVEs should have proof-of-concepts & accurate risk levels.	[85]	35	0.309
	Reflect accountability	CVEs reflect a sense of maturity & attention to security.	[39]	17	0.190
	CVEs are trustworthy	CVEs are the most accurate & reliable disclosure sources.		11	0.174
Overall benefits	Fix prioritization	CVEs help prioritize fixes, e.g., based on their severity.		34	0.169
	Assessing project health	CVEs help determine areas of concern in OSS projects.	[39]	7	0.145
	Transparency	CVEs provide transparency for affected projects.	[13]	25	0.135
	A go-to source	E.g., “the CVE database is a solid resource for me” (P18).		23	0.125
	External expertise	CVEs provide access to security experts & their evaluation.	[5, 11, 13]	18	0.121
	Learning opportunities	CVEs are an avenue of access to security knowledge.	[17, 55]	19	0.105
	Platform integration	CVE-specific features are available in OSS platforms.	[13, 35]	14	0.101
	Patching assistance	CVEs help OSS maintainers develop patches.	[37]	13	0.100
Overall challenges	Producing fixes	CVEs lack sufficient helpful information to lead to a fix.	[76, 80]	34	0.220
	Scoring misalignments	CVE reports do not match practical severity.	[3, 61, 72, 89]	31	0.150
	Limited scope	CVEs only cover publicly disclosed vulnerabilities.	[66]	$-\dagger$	0.140
	Disclosure time lag	Delay in bug discovery & inclusion in the CVE database.	[12, 49, 63]	14	0.118
	Process ambiguity	Maintainers are unsure on what qualifies for a CVE.	[13, 26]	21	0.118
	Can be noisy	CVEs can cause vulnerability fatigue, e.g., too many alerts.	[26, 82]	26	0.105
	Information overload	The content in CVEs is overwhelming.		20	0.080
	Dispute complexity	E.g., hard to simply “reject” or dispute a CVE entry.		9	0.070
Usability challenges	Lack project context	CVEs do not account for unique contexts in OSS projects.	[82, 85]	26	0.170
	CVSS complexity	Difficulty producing accurate & consistent CVSS scores.	[72, 89]	37	0.140
	Poor data quality	CVE entries provide only basic information.	[23, 33]	28	0.127
	Lack of automation	E.g., CVEs are missing automated patch support.	[37]	33	0.121
	Inconsistent formats	There is no consistent CVE ingestion format.	[81]	$-\dagger$	0.118
	CVE process friction	Too much involvement for CVE assignment & follow-ups.	[13, 52]	3	0.113
	Operational overhead	E.g., additional tracking for CVE updates is often manual.	[49]	8	0.109
	Poor communication	Ineffective communication between CVE stakeholders.	[5, 11, 13]	17	0.101
Maintainer wants	Automated CVE creation	Less OSS maintainer involvement for CVE advocacy.	[24, 45]	17	0.132
	Risk assessment support	E.g., context-aware risk assessment of OSS projects.	[74]	36	0.131
	Feedback mechanisms	For reporting poor or low-quality CVEs.		8	0.131
	Accessible resources	Resources for those without a security background.	[11, 13]	29	0.128
	Scoring assistance	Guidance for the vulnerability scoring process.		21	0.122*
	Clear CVE criteria	Guidance for determining whether or not a CVE is needed.	[59, 73]	17	0.122*
	Better communication	Mechanisms for communication between CVE stakeholders.	[5, 11, 13]	10	0.122*
	More centralization	Preference for OSS ecosystem-specific advisories.	[24]	$-\dagger$	0.110

provided in Table 2, where we also include relevant prior work for each factor when applicable. In the following section, we report and discuss the results of 24 semi-structured interviews with OSS maintainers, which are cross-analyzed with survey results, e.g., our listing study. We report quotes as transcribed with minor grammatical corrections and omissions, e.g., using “[...]”, for readability.

The CVE system has been studied extensively, with characterizations of vulnerability propagation and bug-hunting economics, but rarely from the perspective of OSS project maintainers. **We are the first to investigate this perspective, and we provide the first ranking of characteristics’ significance, e.g., CVE usability challenges, derived from maintainers who have advocated**

for or triaged CVE-assigned vulnerabilities. This perspective matters because most maintainers operate without dedicated security training, yet are expected to engage the CVE system, which is designed around vendors and CNAs, with different resources.

We report the number of interview participants who discussed or elaborated on a given topic (*I*), the number of listing study participants who mentioned a factor (*L*), and the relative importance of factors based on Likert-scale worth estimates (π). Such relative importance of factors in a category, e.g., *Overall benefits*, to participants is based on a Likert scale, where a higher π value indicates the probability of one factor being preferred over another, i.e., $\pi \in [0, 1]$, and the sum of worth estimates in a category is 1.

5.1 How CVEs are perceived by OSS maintainers

The nature of the OSS ecosystem encompasses collaboration, transparency, and community. When OSS vulnerabilities are disclosed, OSS maintainers are made aware of these security bugs through various channels, such as Dependabot alerts generated from the GitHub Advisory Database for potentially affected projects [30]. However, when vulnerabilities have an assigned CVE-ID, some interviewees felt they carry additional weight; therefore, CVEs can influence how vulnerabilities are prioritized, triaged, and resolved. **On the criticality and weight of CVEs.** Although not the most mentioned in the listing study, *critical to cybersecurity* is the highest-ranked perception by OSS maintainers ($L = 21$, $\pi = 0.323$). This perception is consistent with *fix prioritization* being the most important benefit ($L = 34$, $\pi = 0.169$), as CVEs help patch priority [16].

During interviews, OSS project maintainers described the importance of ensuring that vulnerabilities with assigned CVEs, or in the process of obtaining a CVE, should be addressed quickly, often ahead of other reported issues. Almost half of the interviewees asserted that CVEs indicate a sense of “urgency” (P6) and should be handled in a timely manner. Further, others mentioned how they feel “like [I am] contributing to the greater good [...] it’s probably giving more priority to fix it” (P12) when a CVE is assigned and is able to propagate through the OSS vulnerability lifecycle to alert other affected projects [12].

Some interviewees also felt that although CVEs are critical to cybersecurity, they carry disproportionate, “unnecessary” (P7) weight compared to security issues without CVEs or those that would not be assigned a CVE. A few interviewees mentioned how they simply do not assign CVEs to reported vulnerabilities because they are only concerned with producing a fix. For instance, an interviewee stated, “I don’t care enough, I don’t need the CVEs for my project” (P7), which is consistent with a recent study that studied how OSS maintainers handle bug bounty reports [11]. However, most interviewees recognize how CVEs serve as a mechanism to catalog and track valid vulnerabilities.

Expectations of rigor. *Should be reproducible* is the most listed perception by OSS maintainers ($L = 35$, $\pi = 0.309$) and ranked as second in the perception category, i.e., OSS maintainers believe it is very important for CVE exploits to be reproducible and have accurate severity accordingly. However, OSS maintainers also mention how *producing fixes* ($L = 34$, $\pi = 0.220$) and *scoring misalignments* ($L = 31$, $\pi = 0.140$) pose the most challenge when consulting CVE information during patch development, raising quality concerns for CVE-assigned OSS vulnerabilities.

Some interviewees described how they believe that consulting CVEs should be sufficient on their own to reproduce security bugs. In particular, an interviewee stated how they “often reproduce the issue myself, but it’s much easier when the CVE includes severity and reproduction steps” (P17). Another interviewee reinforces this idea by describing how “if [a CVE] doesn’t include a risk level, we can’t assess priority” (P4). These perspectives underscore OSS maintainers’ expectations that CVEs should meet a baseline of rigor and further highlight how gaps in CVE detail can delay fixes and mislead interpretations of priority.

Some interviewees believe that proof-of-concept and accurate risk levels are not as important for CVEs. An interviewee stated how “proof-of-concept isn’t always necessary, we just check if the patch is relevant to our code” (P2). In addition, a few other interviewees who work with companies or have chosen to commercialize their OSS projects described how they only focus on patching whenever possible to minimize the number of unresolved CVEs that may be found in their project, regardless of rigor. “They will care if a security issue arrives, you know, because they’ll have implications” (P19). Such perspectives reflect a results-first mindset amongst OSS maintainers and a limited value they see in CVE rigor when fixes are the end goal.

Legitimacy and accountability. CVEs *reflect accountability* ($L = 17$, $\pi = 0.190$) and *are trustworthy* ($L = 11$, $\pi = 0.174$) are the least expressed in all three studies regarding CVE perceptions. Despite their central role in the vulnerability disclosure ecosystem, OSS maintainers downplay their perceptions of CVEs as reliable signals of legitimacy or accountability, i.e., CVEs do not necessarily reflect a sense of maturity and attention to security, nor are they the most accurate and reliable disclosure sources.

A few interviewees cited how CVE presence provides credibility. For instance, an interviewee mentioned how they felt “having a CVE shows that our project takes security seriously” (P17). However, some others described how they treat CVEs “with a grain of salt” (P8), reflecting a sense of low expectations for what CVEs may have to offer. An interviewee stated how “some maintainers just ignore CVEs entirely [...] I don’t think that makes them less mature” (P5). These contrasting perceptions of accountability show how maintainers may see CVEs as signals of credibility, “just another issue” (P15), or are noisy.

A few interviewees argue that CVEs are the most accurate and reliable sources of disclosure for OSS vulnerabilities. One of them stated that they “usually trust the CVE database more than community forums or random GitHub issues” and that “[CVEs] are the closest thing we have to an official source of truth” (P24). This perception of CVEs from OSS maintainers is underrepresented in our studies, highlighting the limited extent to which OSS maintainers regard CVEs as authoritative sources of vulnerability disclosure. Prior vulnerability management research in enterprise and vendor settings [20, 44] generally assumes CVE databases are authoritative by default; our finding that OSS maintainers do not share this assumption suggests that trust in CVEs must be earned through data quality and usefulness rather than institutional backing alone.

5.2 What CVEs enable for OSS maintainers

All three of our studies reflect that most maintainers feel that CVEs provide “an incentive to keep their users safe” (P18), positioning CVEs

as both a motivator for action and transparency for downstream users and dependencies. OSS project maintainers feel it is important to dedicate additional efforts to understand what CVEs have to offer regarding balancing project goals with security, e.g., when a security alert is received on GitHub [30].

Security actionability. As mentioned above, study participants resonate most with the perception that CVEs are *critical to cybersecurity* ($L = 21, \pi = 0.323$) as demonstrated in our Likert-scale study. In particular, participants feel that CVEs are most beneficial for *fix prioritization* ($L = 34, \pi = 0.169$) and are particularly helpful for *assessing project health* ($L = 7, \pi = 0.145$). However, participants felt that CVEs are the least beneficial for *patching assistance* ($L = 13, \pi = 0.100$), which is consistent with how participants reported using CVEs for *producing fixes* ($L = 34, \pi = 0.220$) is the most challenging, which we describe in detail in Section 5.3. These results reflect how maintainers can view CVEs less as authoritative records and more as prioritization and project-level evaluation.

Most interviewees described previous experiences in which they used CVEs to prioritize fixes. For instance, an interviewee described how *“I still prioritize them [...] I would fix it, anyway, so that’s not the question, but CVEs give more priority to fixing it, I’ll escalate it.”* (P1). Another interviewee emphasizes this perspective by stating how *“naturally all the other things get lower priority and it takes longer until we can deliver them”* (P16), and a few of whom reference how having sponsorship for their projects immediately shifts focus to prioritizing security patches. On the other hand, a few interviewees felt CVEs are burden-inducing bugs that can be overwhelming, and describe how they *“already have so much on [their] plate, so [CVEs are] kind of low down there”* (P21). These perspectives reflect how, although CVEs provide a sense of weight to prioritize patch development, their symbolic urgency may outpace the resources maintainers have to respond promptly, which is consistent with prior work [12].

A few interviewees also talked about how they use CVEs to assess the health of their own projects and adopt upstream projects as dependencies. An interviewee stated how *“it’s not perfect, but it gives a snapshot of how I treat the security of my projects”* (P21), i.e., timely CVE remediation indicates a healthy project. Another interviewee described how *“when we evaluate new dependencies, the number and age of CVEs is a good health signal”* (P6), indicating how CVEs can provide a heuristic for evaluating the reliability of dependencies. Another interviewee mentioned how *“projects may still have open CVEs because they don’t bother to close them”* (P7), highlighting the difference between closure and maintenance processes without necessarily reflecting poor security practices.

Transparency and usability benefits. The third most beneficial factor for CVEs is *transparency* ($L = 25, \pi = 0.135$), and study participants describe CVEs as the *go-to source* ($L = 23, \pi = 0.125$) for vulnerability information. Prior work finds that security report visibility can help discover vulnerabilities that were previously not noticed by developers [7, 13]. Further, participants felt that *platform integration* ($L = 14, \pi = 0.101$) of CVEs in OSS, i.e., features that allow maintainers to easily advocate for CVEs as part of the triaging workflow, is particularly beneficial. Ayala et al. [13] also find that OSS maintainers appreciate platform security features when they are integrated into development workflows, since this increases usability and advocacy.

Almost half of the interviewees recognize the value of transparency of CVEs, e.g., for prioritizing fixes, and how transparency embraces general OSS principles, keeping users up to date, and visibility through proper disclosure. An interviewee mentioned how *“I’ve had users thank me for being transparent because I file CVEs [...] downstream users appreciate that they can track known issues through CVEs”* (P22), indicating responsibility for disclosure practices that align with broader OSS principles of openness. Another interviewee reinforced this concept by stating how *“the presence of CVEs does not mean that a project is unhealthy, sometimes it just means that they are transparent”* (P23). On the other hand, most interviewees expressed frustrations with how *“just listing CVEs doesn’t make things transparent, context is always missing”* (P8), perspectives which we dive deeper into in Section 5.4.

Some interviewees also expressed using the CVE database as a go-to source and described positive experiences with using CVE-specific platform features while triaging vulnerabilities. An interviewee described the CVE database as *“a centralized place to verify whether an issue is legit [...] it’s the first place I cross check if something’s flagged in a scan”* (P3). Another interviewee mentioned how *“it’s nice when GitHub automatically links a CVE to a security advisory [...] it puts it straight into our workflow of here’s the advisory, and we do the work in the advisory”* (P4). However, some interviewees feel that *“the integrations are inconsistent, some tools surface CVEs, others bury them or mislabel them [...] dependency transparency has improved more from SBOMs [software bills of materials]”* (P19). This complements prior work where OSS maintainers found limitations of OSS security features, e.g., noisiness [13].

A source of security expertise and learning. Participants ranked *external expertise* ($L = 18, \pi = 0.121$) and *learning opportunities* ($L = 19, \pi = 0.105$) as the fourth and fifth most beneficial factors of CVEs, respectively. In particular, OSS maintainers feel that CVEs provide access to security experts and their evaluation of vulnerabilities, and an avenue of access to security knowledge. Prior work shows that this factor is also the case for bug bounty reports [11].

Most interviewees extensively describe using CVEs as a means to gather a security expert’s perspective of project vulnerabilities or as a learning opportunity. An interviewee described their experience *“having that external review helps validate whether something really is a security issue and is a way for us to benefit from people who specialize in security, even if I don’t have that expertise internally”* (P8). Another interviewee stated how they *“can study how my past vulnerabilities and my project were fixed [...] I’ve learned more from CVEs than I have from most security tutorials”* (P16). A few others described negative experiences regarding CVEs, such as *“automated scanners submitting CVEs, there’s no expert behind it”* (P5). The educational value of CVEs for OSS maintainers depends on the quality (Section 5.4) and depth of the information they provide, not just their existence as identifiers.

5.3 CVE limitations in OSS practice

Although OSS maintainers express how CVEs shine in fix prioritization and providing a metric in which OSS projects handle and disclose vulnerabilities, most of the participants in our three studies expressed general frustration with the development of patches and misalignments of the severity score for CVE-assigned vulnerabilities. OSS maintainers also feel that CVEs often make routine

vulnerability management more cumbersome than needed, introducing administrative overhead and sometimes distracting from implementing timely fixes.

Developing patches. Participants felt that the use of CVEs for *producing fixes* ($L = 34, \pi = 0.220$) is the most overall challenging factor, which is consistent with how participants reported how CVEs are the least beneficial for *patching assistance* ($L = 13, \pi = 0.100$). Half of the participants expressed frustration with how CVEs do not contain relevant potential fixes or only basic information. Prior work has acknowledged automated CVE analysis challenges [4]; we discuss how maintainers deal with such challenges.

Most interviewees described past experiences in triaging “ambiguous” (P21) CVEs that lack fixes relevant to their projects. An interviewee explained how their team has “seen CVEs that just point to a commit without a patch or mitigation” (P13). Another interviewee stated how “many CVEs say what the problem is but not how to fix it [...] it would be great if CVEs included recommended fixes” (P5), further reflecting the lack of suggested fixes or information gaps that are meant to take place during the CVE Record Lifecycle [51]. On the other hand, a few interviewees expressed how CVEs are not always meant to provide a direct solution, but help guide towards a solution to the vulnerability, e.g., “the CVE should describe the vulnerability, not solve it for you” (P10). OSS maintainers view CVEs as awareness signals, though they express that sometimes CVEs are insufficient on their own for full remediation. Prior OSS bug bounty and advisory work [11, 12] does not surface fix production as a prominent maintainer concern; its position as our top-ranked overall challenge, alongside patching assistance as the lowest-ranked benefit, indicates that the CVE system’s separation of disclosure from remediation is a primary friction point for maintainers, most of whom report no security background.

Inaccuracies and noise. Participants report that the second most overall challenging factor for CVEs is *scoring misalignments* ($L = 31, \pi = 0.150$), where the severity reported in CVEs is deemed incorrect by OSS maintainers for practical severity. Furthermore, participants reported that CVEs *can be noisy* ($L = 26, \pi = 0.105$), the third most listed challenge, and can cause *information overload* ($L = 20, \pi = 0.080$), i.e., the content of CVEs is overwhelming, as the sixth and seventh most challenging, respectively. In prior work, dependency alerts with CVEs are known to be potentially noisy [82], and overstated severities have been reported that (1) studied OSS bug bounty reports [11] and (2) revealed concerns about low-quality or noisy reports from bug hunters [7]. Most interviewees also indicated the need for accurate and automated risk assessment.

Some interviewees in our study expressed concerns about overstated severities reported in CVEs. For instance, an interviewee described how their team has “had low-risk bugs get CVSS 9.8 scores just because of hypothetical attack paths [...] this is really something like at most 5.1” (P24). In addition, a few others felt that “users see a high severity and assume the worst [...] it causes unnecessary panic” (P7), highlighting how inflated CVE severity scores can distort user perception of practical impact. While CVSS score inflation has been documented in NVD-level analyses [89], our findings reveal the downstream human cost of OSS maintainers having to manage user panic and pressure from scores they had no role in producing.

However, a few interviewees shared a similar perspective to how initial CVE scoring “is still a useful starting point for prioritization [...] we don’t treat the score as gospel” (P1), describing such scores as an indication for fix prioritization, described in Section 5.2, and how it is accepted by maintainers as not the ground truth; thus reflecting the least agreed perception that *CVEs are trustworthy*.

Some interviewees also described their past experiences with CVEs as noisy and containing too much information. In particular, an interviewee shared a story about their co-maintainer, “but the like one million projects that use [redacted] got a bunch of CVE noise and the maintainer, who like a couple of years later burned out and like rage deleted his GitHub account, ended up having to publish a patch to an old version of [redacted] just to shut up the warnings” (P22). Another interviewee stated, “CVEs often come with too much boilerplate or jargon [...] sometimes [CVEs] are so verbose that the actual issue gets buried” (P24). These same interviewees also reference using CVEs through dependency alerts, e.g., from Dependabot [30], but are dissatisfied with the lack of concise, actionable detail. A few others shared how they “prefer more context, but most CVEs I’ve seen are pretty short” (P2).

Scope and process challenges. OSS maintainers rank *limited scope* ($L = 0, \pi = 0.140$)—i.e., CVEs only cover publicly disclosed vulnerabilities—as the third most overall challenging factor in our Likert-scale study. Although this challenge was not reported in our Listing study, some interviewees argue the importance of publicly disclosing security patches and advocating for CVEs whenever possible so that downstream projects are alerted. In addition, our participants report *disclosure time lag* ($L = 14, \pi = 0.118$) and *process ambiguity* ($L = 21, \pi = 0.118$) in CVE disclosure as the fourth and fifth most overall challenging factors, respectively. Lastly, participants rank *CVE dispute complexity* ($L = 9, \pi = 0.070$) as the least challenging factor when dealing with CVEs.

Prior work has explored identifying silent security patches just in time when they are released for ensuring robust and secure maintenance practices in OSS [77]; however, some interviewees described instances and expressed frustration in which other maintainers quietly fix vulnerabilities and do not disclose them — “I know other maintainers patch security issues that never became CVEs because they chose private remediation [...] that’s a gap” (P9). Further, interviewees also described ambiguity around what qualifies for a CVE. One of whom stated, “I again, going back to that thing of like what’s valid? What’s not valid, right? Like someone being able to digest an issue, then assess a recommendation [...] it takes a lot of the thought process and ambiguity” (P21). Some others discussed how automated CVE assignment, e.g., from an upstream CNA such as huntr, which initiates the CVE assignment process on behalf of maintainers for validated vulnerabilities, is preferred since “there’s no ground truth checklist or standard guidance, it’s mostly guesswork [...] my general rule is user-focused and practical security impact” (P14). Half of the interviewees also expressed a need for clear CVE criteria, which we explore in Section 5.5.

Some interviewees also felt that the disclosure time lag is particularly challenging to deal with during the CVE triaging pipeline. In particular, an interviewee stated, “sometimes I patch an issue and weeks go by but the CVE didn’t show up until much later” (P22). Recent work has revealed how such bottlenecks occur during Step

6 of the CVE Record Lifecycle, where CNAs are required to submit the minimum required data elements are included in the CVE Record so that it is published to the CVE List [12, 51]. However, some interviewees insisted on how *“honestly, I care more about our GitHub advisory than when the CVE goes live”* (P1), challenging the expectations of having vulnerabilities centralized in CVEs.

Although our participants see CVE dispute complexity as the least challenging factor of triaging CVE-assigned vulnerabilities, some interviewees shared detailed opinions on the matter. An interviewee shared their experience, *“I have no idea who to contact or how to appeal [...] there’s no clear mechanism to say, this doesn’t apply, and now this CVE is permanent in the CVE database”* (P20). The ability to successfully appeal a CVE, e.g., because it is inaccurate, remains an ambiguous process for a few interviewees.

5.4 Usability and operational frictions

OSS maintainers also express usability and operational frictions during and after the CVEs Record Lifecycle, i.e., when playing an advocate role for CVEs or a recipient of dependency alerts containing CVEs. This includes, but is not limited to, computing CVSS scores for CVEs, communicating with relevant stakeholders, and triaging CVEs that may be deemed irrelevant or low-impact.

Lacking project contexts and data inconsistencies. Participants in our studies cite *lack project context* ($L = 26, \pi = 0.170$) and *CVSS complexity* ($L = 37, \pi = 0.140$) as the most challenging usability factors when triaging CVE-assigned vulnerabilities. These factors are consistent with the third-ranked CVE usability challenge in our Likert-scale study, *poor data quality* ($L = 28, \pi = 0.127$), and further need for risk assessment support ($L = 36, \pi = 0.131$) and scoring assistance ($L = 21, \pi = 0.122$). OSS maintainers feel that the current state of CVEs does not account for unique contexts for their projects, and question the quality of CVEs and their assessed impact. Wermke et al. [82] report how vulnerabilities in dependencies, e.g., transitive dependency CVEs, often turn out to be false positives or irrelevant for participants’ projects.

Half of the interviewees in our study described previous experiences with triaging unclear or low-quality CVEs for their projects. For instance, an interviewee recalled that *“the CVE just said remote code execution, like what are we supposed to do with that? It wasn’t a super detailed report as I’d like it to be [...] if it’s remote code execution it’s always seems to be considered massively severe which I get, but then contextually within the actual application in my project it doesn’t end up being severe”* (P4). Furthermore, this interviewee later discussed how *“sometimes [CVEs] ignore context like whether it’s behind auth, or if it’s even reachable in real deployments, it’s frustrating”* (P4). Some interviewees shared an opposite perspective—*“CVEs can’t be custom-fit to every project [...] we create an advisory when something seems off. If people need more detail, then they should read that instead”* (P2). These contrasting perspectives reflect (1) how maintainers balance the need for contextual accuracy with the expectation that CVEs remain general-purpose identifiers and (2) serve as motivation for discourse regarding vulnerability centralization. Prior vulnerability management research has focused on context loss at the NVD or scanner level [4, 45]; our finding elevates this context loss to the top usability barrier because OSS projects vary in deployment context, which is a dimension the CVE system was not designed to capture.

Some interviewees also deemed CVSS calculations, e.g., producing accurate and consistent CVSS scores for CVE-assigned vulnerabilities, as difficult to use. An interviewee described how they *“always have issues with calculating a CVSS score [...] the vector strings are cryptic and subjective, you can make a case for multiple scores.”* (P15). Prior work has argued that CVSS scores are still inadequate for both their intended purpose and as a proxy for risk to a vulnerable system [72]. A few other interviewees share a contrasting perspective—*“we use the calculator as a guide, the complexity hasn’t been a major blocker [...] it’s not perfect, but [CVSS] gives us a consistent rubric across projects”* (P23). Although maintainers have mixed experiences with the calculation and interpretation of CVSS, it persists as a default mechanism.

Too much manual effort. The *lack of automation* ($L = 33, \pi = 0.121$) was the fourth-ranked usability challenge for CVEs in our Likert-scale study. Furthermore, participants felt that CVEs have *inconsistent formats* ($L = 0, \pi = 0.118$), i.e., there is no consistent CVE ingestion format, which was the fifth most challenging usability factor in our Likert-scale study. Although this factor was not mentioned in our listing study, interviewees brought attention to how there is no standardized structure among security advisories, widely adopted OSS vulnerability databases, e.g., OSV.dev [58], or tooling for CVEs; however, participants in our Likert-scale study felt more centralization was the least important need.

Despite the availability of platform security features across platforms that support CVE assignment, e.g., to an upstream CNA, some interviewees felt that there is a lack of automation when triaging CVEs. An interviewee stated, *“we patch dozens of CVEs a year [...] we end up Googling or digging through commits just to figure out what to do, why the hell isn’t that automated?”* (P4). A few of these interviewees implemented custom tooling so that their team automatically retrieves upstream patches if available, *“we built our own scripts to pull in upstream patches for CVEs, but it’s brittle [...] automation would save us weeks”* (P20). Furthermore, these interviewees argued that inconsistent CVE formats make it difficult to ingest CVEs for custom tooling.

However, some interviewees were skeptical of using automation methods to triage CVEs. For instance, an interviewee described how *“automating CVE patching sounds great until you break the prod branch [...] CVEs don’t capture all the context, as intended, so you still need human judgement”* (P16). Another interviewee warned against full automation, *“if every CVE auto-patched, you’d have chaos [...] dependencies don’t all move at the same pace so not every fix applies cleanly”* (P23). In addition, a few interviewees pinpointed how inconsistent formats are downstream tooling problems and not CVE problems. Overall, these perspectives highlight that while automation could improve CVE usability by reducing busy work, reliability suffers without human oversight.

Systemic burdens. Participants in our three studies describe *CVE process friction* ($L = 3, \pi = 0.113$), *poor communication* ($L = 17, \pi = 0.101$), and *operational overhead* ($L = 8, \pi = 0.109$) as the least challenging usability factors when triaging CVEs. In particular, OSS maintainers feel that there is too much involvement during and after CVE assignment, tracking, and follow-up communication between relevant stakeholders, e.g., dependent projects.

A few interviewees described the frictions of the CVE process and operational overhead in detail, e.g., *“GitHub is actually worse in*

that aspect because you have to go to the security issue, request a CVE, it takes a couple of minutes, or it could take even days. And then you get the CVE, and then you can work the issue, close it, and patch it, it's a very complicated process" (P21). Another interviewee recounted their challenge with tracking, "we thought once a CVE is issued it would go to the NVD or something, instead, I'm chasing updates across three or four systems [...] we use a separate spreadsheet just to track CVEs and their status" (P9). Regarding CVE assignment, this particular interviewee, i.e., whose project is its own CNA, stated how "CVE tooling is minimal, we're basically emailing back and forth [...] it feels like filling out your taxes" (P9). In contrast, a few interviewees noted that although the process can be tedious, they see these steps as a necessary safeguard to ensure accuracy and accountability. One maintainer explained that "it also forces us to document properly, slow down enough to validate the patch and schedule a release" (P18). In summary, although some OSS maintainers see CVE processes as overhead, others value them as safeguards for accuracy and accountability.

A few interviewees also expressed frustration about ineffective communication with CVE stakeholders. An interviewee recalled, "we sent questions to our CNA and didn't hear back for weeks [...] no one tells you what happens after you submit, it just disappears" (P12). Prior work situates such friction inside organizations, e.g., complex systems, poor defaults, unclear internal responsibilities [20]. The friction here is external, occurring at the boundary between a project and CNAs, MITRE, and the NVD, where maintainers have no standing and no escalation path [57].

On the other hand, another interviewee described relatively positive experiences with the CVE process, "we've never had issues communicating with MITRE or the CNA once we knew the process, even when things came up that we needed to fix or address" (P6). A few other interviewees also shared this perspective. The varying communication experiences between OSS maintainers and stakeholders are inconsistent, but these interactions could shape trust in CVE reporting and triaging for future reported vulnerabilities.

5.5 CVE factors desired by OSS maintainers

Despite the importance of understanding CVE workflows and severity scoring, given how Dependabot is the primary method of alerting affected projects about security updates on GitHub [29], most of the OSS maintainers in our studies do not have a security background. Lastly, we asked participants about additional features or improvements to reduce additional work while creating, triaging, and following up with CVE-assigned vulnerabilities that would benefit their current approaches the most.

Decision support. Our participants expressed the greatest desire for mechanisms that support *automated CVE creation* ($L = 17, \pi = 0.132$) when triaging security reports to reduce their involvement for CVE advocacy to upstream CNAs, e.g., GitHub Security Lab. In addition, participants also want *clear CVE criteria* to determine whether or not a CVE is needed. Overall, OSS maintainers described these wants as essential to (1) streamline vulnerability disclosure and (2) reduce the manual burden of CVE advocacy.

Half of the interviewees in our study argued for more automation during the CVE advocacy process. For instance, an interviewee stated, "I wish security bugs could just be detected and boom, CVE assigned and sent off [...] this would be a dream, we could just confirm

and move on to all the other things we have on our plates" (P9). In addition, some interviewees felt that CVEs are a very subjective method of evaluating the impact of vulnerabilities—"I have no idea what like objectively qualify something is worth having a CVE, I don't even know what CVE stands for, sometimes we ask other maintainers because it's not clear at all [...] if somebody asks nicely I'll advocate for it" (P15).

In contrast, a few interviewees felt that more automation is not the answer, e.g., "automatic CVEs would just add more noise, I really don't think every security issue deserves one [...] we'll be drowning in low-impact CVEs nobody takes seriously" (P5). Furthermore, a few interviewees argued that CVEs are not necessary for small projects, one of whom stated, "my stuff isn't that widely used? So it's like, okay, it doesn't really matter if there's a vulnerability in my code because it's just a project that [I] did on the weekend" (P17). Regarding clearer CVE criteria, a few interviewees advise other maintainers to "just use your best judgment, not everything needs a rulebook [...] when you're so busy juggling all these issues around, just follow your gut" (P14). OSS maintainers feel that although automation and clearer CVE criteria could ease the burden of CVE advocacy, they also risk introducing noise and oversimplifying nuanced decisions that will hinder valid vulnerabilities from reaching maximum awareness and timely disclosure. Prior work on automated vulnerability handling has emphasized the technical feasibility of detection and repair [24, 87]; our participants' desire for automation is driven less by detection accuracy than by the administrative burden of CVE advocacy and unclear assignment criteria, a friction reported even by a maintainer whose project is itself a CNA (P9).

Scoring support and resources. In our Likert-scale study, participants ranked *risk assessment support* ($L = 36, \pi = 0.131$) the second most wanted factor when triaging CVEs, followed by *accessible resources* ($L = 29, \pi = 0.128$) and *scoring assistance* ($L = 21, \pi = 0.122$) as the fourth and fifth ranked wants, respectively. In previous work, maintainers have mentioned similar needs with respect to OSS-centered platform security features [13], e.g., assisted triaging and accessible resources for vulnerability management; further, researchers have questioned the use of CVSS to evaluate vulnerabilities, citing that the CVSS scoring algorithm is not justified, either formally or empirically [72].

Half of the interviewees in our study want context-aware risk assessment of their projects when receiving or advocating for a CVE and guidance to produce an accurate severity score. For example, some interviewees shared the perspective of how "it would help if the scoring factored in the practical perspective, like is this code path even reachable in most builds? [...] we need tools that tell us does this vulnerability actually put our users at risk?" (P23). Another interviewee stated, "on like a gut level, I was like, this doesn't seem like a critical problem to me, I also didn't really feel like I had that clear grasp on how to produce those ratings [...] either of the reality of the issue or of the like objective rating that it should have according to that scale" (P17). On the other hand, a few other interviewees prioritized patching over precise severity scores—"we just patch and move on, wasting time on debating context feels like overkill when we're already low on manpower [...] when we have to [calculate CVSS scores], we just click the boxes and move on. And if somebody yells at me for it, I say 'no', it's bullshit anyways. Go away, I'm trying to get work done" (P19). These views demonstrate how maintainers

are split between valuing context-aware scoring for accuracy and dismissing it in favor of quick patching.

Most interviewees extensively discussed the need for accessible resources when reviewing and patching CVE-assigned vulnerabilities. An interviewee explicitly stated how *“we need more ‘explain like I’m five’ versions of CVE, or really just security in general since there’s so much to dig through”* (P3). Another interviewee, who self-reported having a security background, stated how *“you really have to develop a security mindset, like moving forward, having secure coding practices in mind. Ask yourself, does this vulnerability really affect my project? The how and why are the most important”* (P7).

Improved feedback loops and centralization. Participants in our three studies mentioned the need for improved *feedback mechanisms* ($L = 8, \pi = 0.131$) with respect to poor or low-quality CVEs and *better communication* ($L = 10, \pi = 0.122$) between relevant CVE stakeholders, e.g., upstream CNAs. Although not identified in our listing study, some interviewees discussed the need for *more centralization* ($L = 0, \pi = 0.110$), i.e., tailored for the OSS ecosystem. OSS maintainers felt that having such feedback mechanisms and communication would strengthen trust in the CVE process, clarify decision-making, and provide transparency they could rely on during CVE triage. Further, maintainers challenge the notion of using CVEs and instead having centralization for OSS vulnerabilities.

A few interviewees argued for improved feedback mechanisms for reporting poor and low-quality CVEs. In addition, a few interviewees referenced negative or non-productive interactions with CVE stakeholders. An interviewee reported, *“right now, you just have to email someone and hope they care [...] it’s unclear who to talk to when something goes wrong or needs to be updated. MITRE? The reporter? NVD?”* (P8). Some others recalled positive experiences or workarounds when mishaps arise—*“we’ve had no trouble contacting folks, maybe it depends on your CNA, but those kinds of things are pretty easy to sort out [...] if a CVE is just plain wrong, we ignore it and mark it as resolved to silence the alert”* (P16). Maintainers who work with responsive CNAs or go-to workarounds feel less burden when dealing with inaccurate CVEs, while others may have additional concerns and lack reliable channels for redress, reflecting inconsistent feedback loops and transparency in the CVE process.

Some interviewees expressed a preference for the centralization of OSS vulnerabilities to streamline resolution workflows. An interviewee stated, *“I don’t care enough about [the CVE process], I don’t need the CVEs for my project [...] I don’t need more work on my table, if somebody else assigns CVEs for things in our issue tracker, fine, we’ll get to those. Platform advisories like GitHub’s feel more integrated anyway, hell, submit it to public issues if you want”* (P11). Another interviewee argued how *“if it was up to me, I wouldn’t bother with CVEs at all, we would just publish advisories on GitHub and call it a day since our users follow that, not CVE IDs [...] using CVEs feel like forcing a square peg into a round hole, we recently stopped bothering with CVEs for minor issues”* (P9). Some other interviewees shared the perspective of how *“sure, CVEs are annoying, but they make our vulnerabilities visible to the right people [...] CVEs reach security scanners, vendors, and downstream packagers, you name it”* (P3). These perspectives challenge centralization against decentralization for OSS ecosystem-specific advisories and the role of CVEs as universal identifiers, reflecting tensions between project workflows and ecosystem visibility.

6 Discussion

6.1 Challenges and their implications

OSS project maintainers mentioned how CVSS calculations are difficult to produce accurately and consistently, as mentioned the most in our Listing survey study ($L = 37$). In addition, they ranked the generation of fixes as the most challenging overall, noting that CVEs often lack relevant potential solutions and do not consider the unique contexts of OSS projects ($I = 12$), in our Likert-scale survey study when triaging CVEs ($\pi = 0.220$). Other relevant challenges include scoring misalignments between CVE severities and practical impact, and poor CVE data quality, issues that compound the top challenges our studies identify. These maintainer-centered challenges highlight (1) how CVEs fail to adapt to the unique needs in OSS and (2) the need to further explore short-term action items and long-term plans that can help align CVEs closely with OSS.

CVSS calculations and CVEs missing contexts of specific OSS projects overwhelm most OSS maintainers in our studies ($L = 51$) and take away from time spent developing fixes. These strains highlight opportunities to enhance CVSS with targeted explanations, e.g., project-aligned versions for maintainers during the triaging process, and investigate how generative AI can automate producing such explanations to make severity assessments more actionable. Despite the availability of context-aware and severity impact analysis tools [3, 72], their limited use among OSS maintainers underscores a gap between technical capability and practical adoption during the CVE triaging process. OSS platforms should integrate context-aware analysis tooling within existing security workflows to facilitate such adoption, e.g., within security advisories, to streamline contextual relevance and reduce redundant triage work for OSS maintainers. The CVE program and similar vulnerability databases should combine automated validation with curator oversight to address such poor quality issues in “not so hot” (P18) CVEs and provide CVE triagers, e.g., maintainers, the ability to rate CVE clarity and impact accuracy to help identify noise.

Other prominent challenges OSS project maintainers faced during CVE creation and triaging are poor data quality, lack of automation, and ambiguity and friction of the CVE process. OSS platforms should implement mechanisms that can automatically identify complex or low-quality CVEs and make them digestible for OSS maintainers without a security background. OSS platforms should also provide OSS project maintainers with the ability to feasibly escalate concerns, request clarifications, or contest inaccurate CVE assignments within advisory mechanisms, e.g., as part of managing GitHub security advisories [28].

6.2 Benefits and their implications

Fix prioritization, project health assessment, and transparency were amongst the most important benefits for OSS project maintainers when triaging CVEs and considering adopting software from the broader OSS ecosystem. Other relevant benefits include security-oriented learning opportunities and recognition of how CVEs can be essential for timely and effective vulnerability management.

OSS project maintainers should communicate the positive impact of proactively engaging with CVEs so that other projects reluctant to proceed with CVE advocacy or triaging can be persuaded by the benefits of shared visibility, coordinated remediation, and improved

ecosystem trust in CVEs. We discourage OSS project maintainers from keeping patched vulnerabilities private, which is, in fact, an already discouraged practice [27, 36], and should instead take advantage of integrated advisory mechanisms that promote vulnerability awareness regardless of project size or perceived maturity to prevent downstream users or projects from unknowingly using vulnerable versions. Further, sharing patches with maintainer-curated context also helps the broader ecosystem learn unique use cases and helps prevent or fix future similar flaws. OSS platforms should consider implementing incentive structures that reward maintainers for timely engagement with CVEs and responsiveness to vulnerability disclosures. For instance, such platforms could award badges, e.g., *CVE Response Leader* or *Rapid Patcher*, to maintainers who resolve CVEs within recommended timeframes, e.g., 7 days for critical CVEs. The CVE program should consider implementing features that allow maintainers to see and share CVE engagement metrics, e.g., improved fix turnaround or visibility to downstream users, to associate CVE triaging with positivity.

CVE triaging also brought about OSS maintainers with valuable learning opportunities. OSS maintainers should approach CVE triaging as a way to learn how vulnerabilities manifest in real-world contexts and use such experiences to refine their long-term maintenance and disclosure practices. OSS platforms should consider providing guided triage workflows, contextual explanations, and automated recommendations where possible, so that OSS maintainers can better interpret CVE data and severity metrics, allowing for more informed patch development and enabling streamlined fixes. The CVE program and vulnerability database maintainers should consider curating well-triaged cases for OSS maintainers, and CVE users in general, to facilitate community-driven learning from real-world experiences.

6.3 Cross-category analysis and future work

Our studies highlight that manual triage efforts, combined with the difficulty of interpreting and validating CVE information for scoring misalignments, identifying impact based on project context, and producing fixes, emphasize the need to automate parts of the CVE triaging pipeline to ease the burden on maintainers. Together, these factors reveal that without impact assessment, CVEs can seem more like distractions and additional validation work as a result of noisy or incomplete data from upstream. Prior research shows that maintainers in enterprise settings face internal friction, e.g., complex systems, poor defaults, and unclear responsibilities [20]. Our findings indicate that OSS maintainers exhibit evidence of external friction, e.g., from outside CNAs and platform processes. Although CVEs are designed to simplify risk mitigation by standardizing vulnerability information, OSS project maintainers' primary pain points stem from generic CVE data failing to meet project-specific needs. Our studies further indicate an unmet need to develop and evaluate infrastructure and feedback mechanisms that address inefficiencies in coordination, response timeliness, and maintainer support during the CVE triaging process.

Our study suggests a need to explore further (1) why OSS maintainers are not using tools that support context-aware analysis, and (2) what barriers prevent their adoption. Notably, no participant reported using AI in a CVE-related workflow or spontaneously

proposed AI as a solution, even though producing fixes and evaluating vulnerability impact are activities for which AI assistance may appear technically applicable [60, 75, 90]. This absence indicates that the open research problem is not only whether AI can perform these tasks, but whether maintainers would recognize, trust, and adopt such assistance within existing vulnerability management workflows. Moreover, recent community reports of AI-generated vulnerability submissions that are inaccurate or low-quality suggest that AI may simultaneously be a source of CVE noise and a potential remedy, making trust calibration an even more pressing design challenge for future tooling [25]. Accordingly, AI support should be evaluated as human-in-the-loop decision support rather than assumed to be a desired replacement for maintainer judgment. Maintainers rank "fix prioritization" as the top benefit, "patching assistance" is the lowest-ranked benefit, "producing fixes" is the most challenging, and "lacking project context" is the top usability barrier. Prior studies on bug bounties and advisories discussed little-to-no concerns about fix prioritization, patching assistance, or lacking project context, while producing fixes was, at most, a middling challenge [11, 12].

We also encourage researchers to develop and share resources catered towards developers who lack a security background that aim to scaffold the CVE triaging lifecycle for OSS maintainers; we plan to further uncover and work towards addressing CVE knowledge gaps by using AI programming assistants for guided and gamified learning, e.g., with scenario-based challenges, since researchers have explored how LLMs can effectively support personalized learning and security awareness initiatives [8]. Our findings uniquely contribute to this effort by surfacing the lived experiences and unmet needs of OSS maintainers that can help identify concrete learning objectives for future AI-driven OSS security games.

7 Conclusion and RQ takeaways

In this paper, we conduct a mixed-methods study, i.e., two surveys and semi-structured interviews, to identify and quantify the factors that affect how OSS maintainers create, perceive, and triage CVEs. We find that maintainers perceive CVEs as critical to cybersecurity, yet downplay them as reliable signals. This dynamic is pronounced in OSS: Maintainers see external expertise and learning opportunities as benefits because they lack the security backgrounds the CVE system assumes, yet the system provides neither the project context nor the scoring guidance to translate urgency into action (RQ1). The factors maintainers value most (e.g., fix prioritization) depend on accurate metadata (e.g., severity scores), yet their biggest challenges (producing fixes, scoring misalignment, lack of project context) arise when that metadata is inaccurate or generic. Despite "lacking project context" being the top usability barrier and "CVSS complexity" the most listed complaint, all of which AI may assist with, zero participants mentioned using AI for CVE-related workflows (RQ2). We include all survey and interview instruments used, as well as an extended ethics discussion in our artifact [1].

References

- [1] 2026. Repository for open science. <https://doi.org/10.5281/zenodo.22262385>.
- [2] Krishnashree Achuthan, Sreeikutty SudhaRavi, Renuka Kumar, and Raghu Raman. 2014. Security vulnerabilities in open source projects: An India perspective. In *2014 2nd International Conference on Information and Communication Technology (ICICT)*. 18–23. doi:10.1109/ICICT.2014.6914033

- [3] Manuj Aggarwal. 2023. A Study of CVSS v4.0: A CVE Scoring System. In *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)*, Vol. 6. 1180–1186.
- [4] Ehsan Aghaei, Ehab Al-Shaer, Waseem Shadid, and Xi Niu. 2023. Automated CVE Analysis for Threat Prioritization and Impact Prediction. *arXiv* (2023). arXiv:2309.03040 [cs.CR] <https://arxiv.org/abs/2309.03040>
- [5] Omer Akgul, Taha Eghtesad, Amit Elazari, Omprakash Gnawali, Jens Grossklags, Michelle L Mazurek, Daniel Votipka, and Aron Laszka. 2023. Bug hunters' perspectives on the challenges and benefits of the bug bounty ecosystem. In *32nd USENIX Security Symposium (USENIX Security)*, Vol. 2301. doi:10.48550/arXiv.2309.03040
- [6] Cloud Security Alliance. 2024. A Vulnerability Management Crisis: The Issues with CVE. <https://cloudsecurityalliance.org/blog/2024/11/21/a-vulnerability-management-crisis-the-issues-with-cve>.
- [7] Noura Alomar, Primal Wijesekera, Edward Qiu, and Serge Egelman. 2020. "You've got your nice list of bugs, now what?" vulnerability discovery and management processes in the wild. In *Proceedings of the Sixteenth USENIX Conference on Usable Privacy and Security (SOUPS'20)*. USENIX Association, USA, Article 18, 21 pages.
- [8] Hany F. Atlam. 2025. LLMs in Cyber Security: Bridging Practice and Education. *Big Data and Cognitive Computing* 9, 7 (2025). doi:10.3390/bdcc9070184
- [9] Autobahn Security. 2025. CVE Database Crisis: What Every Security Professional Must Know. <https://autobahn-security.com/blog/cve-database-crisis/>.
- [10] Jessy Ayala and Joshua Garcia. 2023. An Empirical Study on Workflows and Security Policies in Popular GitHub Repositories. In *2023 IEEE/ACM 1st International Workshop on Software Vulnerability (SVM)*. doi:10.1109/SVM59160.2023.00006
- [11] Jessy Ayala, Steven Ngo, and Joshua Garcia. 2025. A Deep Dive into How Open-Source Project Maintainers Review and Resolve Bug Bounty Reports. In *025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 522–538. doi:10.1109/SP61157.2025.00063
- [12] Jessy Ayala, Yu-Jye Tung, and Joshua Garcia. 2025. Investigating Vulnerability Disclosures in Open-Source Software Using Bug Bounty Reports and Security Advisories. *arXiv preprint arXiv:2501.17748* (2025).
- [13] Jessy Ayala, Yu-Jye Tung, and Joshua Garcia. 2025. A Mixed-Methods Study of Open-Source Software Maintainers On Vulnerability Management and Platform Security Features. In *34th USENIX Security Symposium (USENIX Security 25)*. 2105–2124.
- [14] H.R. Bernard. 2017. *Research Methods in Anthropology: Qualitative and Quantitative Approaches*. Rowman & Littlefield Publishers. <https://books.google.com/books?id=2Fk7DwAAQBAJ>
- [15] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
- [16] Noah Bühlmann and Mohammad Ghafari. 2022. How do developers deal with security issue reports on GitHub?. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 1580–1589. doi:10.1145/3477314.3507123
- [17] Yung-Yu Chang, Pavol Zavarsky, Ron Ruhl, and Dale Lindsog. 2011. Trend Analysis of the CVE for Software Vulnerability Management. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 1290–1293. doi:10.1109/PASSAT/SocialCom.2011.184
- [18] Kathy Charmaz. 2014. Constructing grounded theory. *Introducing Qualitative Methods Series* (2014).
- [19] Ting-Han Chen, Carlotta Tagliaro, Martina Lindorfer, Kevin Borgolte, and Jeroen Van Der Ham-De Vos. 2024. Are You Sure You Want To Do Coordinated Vulnerability Disclosure?. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 307–314. doi:10.1109/EuroSPW61312.2024.00039
- [20] Constanze Dietrich, Katharina Krombolz, Kevin Borgolte, and Tobias Fiebig. 2018. Investigating System Operators' Perspective on Security Misconfigurations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (Toronto, Canada) (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1272–1289. doi:10.1145/3243734.3243794
- [21] Regina Dittrich, Brian Francis, Reinhold Hatzinger, and Walter Katzenbeisser. 2007. A paired comparison approach for the analysis of sets of Likert-scale responses. *Statistical Modelling* 7, 1 (2007), 3–28. doi:10.1177/1471082X0600700102
- [22] Regina Dittrich and Reinhold Hatzinger. 2009. Fitting loglinear Bradley-Terry models (LLBT) for paired comparisons using the R package premod. *Psychology Science Quarterly* 51 (2009), 216–242. <https://api.semanticscholar.org/CorpusID:26576838>
- [23] Ying Dong, Wenbo Guo, Yueqi Chen, Xinyu Xing, Yuqing Zhang, and Gang Wang. 2019. Towards the Detection of Inconsistencies in Public Security Vulnerability Reports. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 869–885. <https://www.usenix.org/conference/usenixsecurity19/presentation/dong>
- [24] Felix Fischer, Jonas Höbenreich, and Jens Grossklags. 2023. The Effectiveness of Security Interventions on GitHub. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (Copenhagen, Denmark) (CCS '23)*. Association for Computing Machinery, New York, NY, USA, 2426–2440. doi:10.1145/3576915.3623174
- [25] Patrick Garrity. 2026. The First CVE Wave: Signs That AI-Assisted Vulnerability Discovery Is Reshaping Disclosure Volumes. VulnCheck Blog. <https://www.vulncheck.com/blog/ai-assisted-vulnerability-discovery>
- [26] Rajdeep Ghosh, Shiladitya De, and Mainack Mondal. 2025. "I wasn't sure if this is indeed a security risk": Data-driven Understanding of Security Issue Reporting in GitHub Repositories of Open Source npm Packages. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security '25)*.
- [27] GitHub. [n.d.]. About coordinated disclosure of security vulnerabilities. <https://docs.github.com/en/code-security/security-advisories/guidance-on-reporting-and-writing-information-about-vulnerabilities/about-coordinated-disclosure-of-security-vulnerabilities#standard-process>.
- [28] GitHub. 2017. About global security advisories. <https://docs.github.com/en/code-security/security-advisories/working-with-global-security-advisories-from-the-github-advisory-database/about-global-security-advisories>.
- [29] GitHub. 2017. Code security documentation. <https://docs.github.com/en/code-security>.
- [30] GitHub. 2019. Dependabot: Automated dependency updates built into GitHub. <https://github.com/dependabot>.
- [31] GitHub. 2026. GitHub terms of service. <https://docs.github.com/en/site-policy/acceptable-use-policies/github-acceptable-use-policies>.
- [32] Barney G Glaser. 1965. The constant comparative method of qualitative analysis. *Social problems* 12, 4 (1965), 436–445.
- [33] Hao Guo, Sen Chen, Zhenchang Xing, Xiaohong Li, Yude Bai, and Jiamou Sun. 2022. Detecting and Augmenting Missing Key Aspects in Vulnerability Descriptions. *ACM Trans. Softw. Eng. Methodol.* 31, 3, Article 49 (April 2022), 27 pages. doi:10.1145/3498537
- [34] Yu Huang, Dena Ford, and Thomas Zimmermann. 2021. Leaving My Fingerprints: Motivations and Challenges of Contributing to OSS for Social Good. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 1020–1032. doi:10.1109/ICSE43902.2021.00096
- [35] Sushawapak Kancharoendee, Thanat Pichitphanphong, Chanikarn Jongyinyong, Brittany Reid, Raula Gaikovina Kula, Morakot Choetkiertikul, Chaiyong Ragkhitwetsagul, and Thanwadee Sunetnanta. 2025. On Categorizing Open Source Software Security Vulnerability Reporting Mechanisms on GitHub. arXiv:2502.07395 [cs.SE] <https://arxiv.org/abs/2502.07395>
- [36] Jessica Kiel. 2025. Patch Management Process: A Guide for Implementation & Best Practices. <https://linfordco.com/blog/patch-management-process/>.
- [37] Seulbae Kim, Seunghoon Woo, Heejo Lee, and Hakjoo Oh. 2017. VUDDY: A Scalable Approach for Vulnerable Code Clone Discovery. In *2017 IEEE Symposium on Security and Privacy (SP)*. 595–614. doi:10.1109/SP.2017.62
- [38] Sagy Kratu. 2025. What Happens When the CVE Program Runs Out of Cash. <https://www.vicarius.io/articles/the-mitre-cve-program-ends>.
- [39] Frank Li and Vern Paxson. 2017. A Large-Scale Empirical Study of Security Patches. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2201–2215. doi:10.1145/3133956.3134072
- [40] Kaixuan Li, Jian Zhang, Sen Chen, Han Liu, Yang Liu, and Yixiang Chen. 2024. PatchFinder: A Two-Phase Approach to Security Patch Tracing for Disclosed Vulnerabilities in Open-Source Software. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (Vienna, Austria) (ISSTA 2024)*. Association for Computing Machinery, New York, NY, USA, 590–602. doi:10.1145/3650212.3680305
- [41] Xiaozhou Li, Sergio Moreschini, Zheyang Zhang, Fabio Palomba, and Davide Taibi. 2023. The anatomy of a vulnerability database: A systematic mapping study. *Journal of Systems and Software* 201 (2023), 111679. doi:10.1016/j.jss.2023.111679
- [42] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. 2018. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. In *Proceedings of the 2018 Network and Distributed Systems Security (NDSS) Symposium*. doi:10.14722/ndss.2018.23158
- [43] Jackson Lim, Yi Lin Lau, Leshawn Khin Ming Chan, Jia Ming Tristan Paul Goo, Huahua Zhang, Zhiyuan Zhang, and Huaqun Guo. 2023. CVE Records of Known Exploited Vulnerabilities. In *2023 8th International Conference on Computer and Communication Systems (ICCCS)*. 738–743. doi:10.1109/ICCCS57501.2023.10150856
- [44] Jiahui Lin, Bram Adams, and Ahmed E. Hassan. 2023. On the coordination of vulnerability fixes: An empirical study of practices from 13 CVE numbering authorities. *Empirical Softw. Engg.* 28, 6 (Nov. 2023), 34 pages. doi:10.1007/s10664-023-10403-x
- [45] Andrea Lisi, Prateeti Mukherjee, Laura De Santis, Lei Wu, Dmitriy Lagutin, and Yki Kortessniemi. 2022. Automated Responsible Disclosure of Security Vulnerabilities. *IEEE Access* 10 (2022), 10472–10489. doi:10.1109/ACCESS.2021.3126401
- [46] Shuhan Liu, Jiayuan Zhou, Xing Hu, Filipe Roseiro Cogo, Xin Xia, and Xiaohu Yang. 2025. An Empirical Study on Vulnerability Disclosure Management of Open Source Software Systems. *ACM Trans. Softw. Eng. Methodol.* 34, 7, Article 214 (Aug. 2025), 31 pages. doi:10.1145/3716822
- [47] Zhuang Liu, Xing Hu, Jiayuan Zhou, and Xin Xia. 2025. From Industrial Practices to Academia: Uncovering the Gap in Vulnerability Research and Practice. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. 129–141. doi:10.1109/MSR66628.2025.00030

- [48] Gitta H. Lubke and Bengt O. Muthén. 2004. Applying Multigroup Confirmatory Factor Models for Continuous Outcomes to Likert Scale Data Complicates Meaningful Group Comparisons. *Structural Equation Modeling: A Multidisciplinary Journal* 11, 4 (2004), 514–534. doi:10.1207/s15328007sem1104_2
- [49] Jing Luo, Heyuan Shi, Yongchao Zhang, Runzhe Wang, Yuheng Shen, Yua Chen, Xiaohai Shi, Rongkai Liu, Chao Hu, and Yu Jiang. 2024. CVECenter: Industry Practice of Automated Vulnerability Management for Linux Distribution Community. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (Porto de Galinhas, Brazil) (FSE 2024). Association for Computing Machinery, New York, NY, USA, 329–339.
- [50] Peter Mell and Miles Tracy. 2002. Procedures for Handling Security Patches. <https://csrc.nist.gov/pubs/sp/800/40/final>.
- [51] MITRE. 1999. CVE Record Lifecycle. <https://www.cve.org/About/Process>.
- [52] Giovane C. M. Moura and John Heidemann. 2023. Vulnerability Disclosure Considered Stressful. *SIGCOMM Comput. Commun. Rev.* 53, 2 (July 2023), 2–10. doi:10.1145/3610381.3610383
- [53] National Institute of Standards and Technology. 2024. How We Assess Acceptance Level. <https://nvd.nist.gov/vuln/cvmap/How-We-Assess-Acceptance-Levels>.
- [54] National Institute of Standards and Technology. 2025. NVD Dashboard. <https://nvd.nist.gov/general/nvd-dashboard>.
- [55] Stephan Neuhaus and Thomas Zimmermann. 2010. Security Trend Analysis with CVE Topic Models. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*. 111–120. doi:10.1109/ISSRE.2010.53
- [56] Viet Hung Nguyen, Stanislav Dashevskiy, and Fabio Massacci. 2016. An automatic method for assessing the versions affected by a vulnerability. *Empirical Softw. Engg.* 21, 6 (Dec. 2016), 2268–2297. doi:10.1007/s10664-015-9408-2
- [57] NIST. 1999. CVEs and the NVD Process. <https://nvd.nist.gov/general/cve-process>.
- [58] OpenSSF. 2021. Open source vulnerabilities database. <https://osv.dev/>.
- [59] Shengyi Pan, Jiayuan Zhou, Filipe Roseiro Cogo, Xin Xia, Lingfeng Bao, Xing Hu, Shanping Li, and Ahmed Hassan. 2022. Automated unearthing of dangerous issue reports. In *30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. 834–846. doi:10.1145/3540250.3549156
- [60] Zhoujin Pan, Jia Liu, Yifan Dai, and Wenjun Fan. 2024. Large Language Model-enabled Vulnerability Investigation: A Review. In *2024 International Conference on Intelligent Computing and Next Generation Networks (ICNGN)*. 01–05. doi:10.1109/ICNGN63705.2024.10871716
- [61] Ivan Pashchenko, Henrik Plate, Serena Elisa Ponta, Antonino Sabetta, and Fabio Massacci. 2018. Vulnerable open source dependencies: counting those that matter. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '18)*. Association for Computing Machinery, New York, NY, USA, Article 42, 10 pages. doi:10.1145/3239235.3268920
- [62] Yangheran Piao, Jingjie Li, and Daniel W. Woods. 2025. Measuring the Vulnerability Disclosure Policies of AI Vendors. arXiv:2509.06136 [cs.CR]
- [63] Gede Artha Azriadi Prana, Abhishek Sharma, Lwin Khin Shar, Darius Foo, Andrew Santosa, Asankhaya Sharma, and David Lo. 2021. Out of sight, out of mind? How vulnerable dependencies affect open-source projects. In *Empirical Software Engineering* 26, 59. doi:10.1007/s10664-021-09959-3
- [64] Haris Pylarinos. 2025. What a future without CVEs means for cyber defense. <https://www.helpnetsecurity.com/2025/05/06/cve-program-foundation/>.
- [65] Sandra Rivera Pérez, Michel Van Eeten, and Carlos H. Gañán. 2024. Patchy Performance? Uncovering the Vulnerability Management Practices of IoT-Centric Vendors. In *2024 IEEE Symposium on Security and Privacy (SP)*. 1198–1216. doi:10.1109/SP54263.2024.00154
- [66] Ralf Ramsauer, Lukas Bulwahn, Daniel Lohmann, and Wolfgang Maurer. 2020. The Sound of Silence: Mining Security Vulnerabilities from Secret Integration Channels in Open-Source Projects. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop* (Virtual Event, USA) (CCSW'20). Association for Computing Machinery, New York, NY, USA, 147–157. doi:10.1145/3411495.3421360
- [67] Jukka Ruohonen, Sami Hyrnynsalmi, Sampsa Rauti, and Ville Leppänen. 2017. Mining social networks of open source CVE coordination. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement* (Gothenburg, Sweden) (IWSM Mensura '17). Association for Computing Machinery, New York, NY, USA, 176–188. doi:10.1145/3143434.3143458
- [68] Johnny Saldaña. 2021. The Coding Manual for Qualitative Researchers (4th edition). Sage (2021), 1–440.
- [69] Pattaraporn Sangaroonsilp, Hoa Khanh Dam, and Aditya Ghose. 2023. On Privacy Weaknesses and Vulnerabilities in Software Systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1071–1083. doi:10.1109/ICSE48619.2023.00097
- [70] Omar Santos. 2025. The Need for a Strong CVE Program. <https://blogs.cisco.com/security/the-need-for-a-strong-cve-program>.
- [71] Murugiah Souppaya and Karen Scarfone. 2022. Guide to Enterprise Patch Management Planning: Preventative Maintenance for Technology. <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r4.pdf>.
- [72] Jonathan Spring, Eric Hatleback, Allen Householder, Art Manion, and Deana Shick. 2021. Time to Change the CVSS? *IEEE Security & Privacy* 19, 2 (2021), 74–78. doi:10.1109/MSEC.2020.3044475
- [73] Yuan Stevens, Stephanie Tran, and Ryan Atkinson. 2021. See something, say something? Coordinating the disclosure of security vulnerabilities in Canada's infrastructure. In *2021 IEEE International Symposium on Technology and Society (ISTAS)*. 1–5. doi:10.1109/ISTAS52410.2021.9629214
- [74] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitras. 2022. Expected Exploitability: Predicting the Development of Functional Vulnerability Exploits. In *31st USENIX Security Symposium (USENIX Security 22)*. <https://www.usenix.org/conference/usenixsecurity22/presentation/suciu>
- [75] Seyed Mohammad Taghavi Far and Farid Feyzi. 2025. Large language models for software vulnerability detection: a guide for researchers on models, methods, techniques, datasets, and metrics. *Int. J. Inf. Secur.* 24, 2 (Feb. 2025), 66 pages. doi:10.1007/s10207-025-00992-7
- [76] Xin Tan, Yuan Zhang, Chenyuan Mi, Jiajun Cao, Kun Sun, Yifan Lin, and Min Yang. 2021. Locating the Security Patches for Disclosed OSS Vulnerabilities with Vulnerability-Commit Correlation Ranking. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) (CCS '21). 3282–3299. doi:10.1145/3460120.3484593
- [77] Xunzhu Tang, Kisub Kim, Saad Ezzini, Yewei Song, Haoye Tian, Jacques Klein, and Tegawende Bissyande. 2025. Just-in-Time Detection of Silent Security Patches. *ACM Trans. Softw. Eng. Methodol.* (July 2025). doi:10.1145/3749370
- [78] Richard J. Thomas, Joseph Gardiner, Tom Chothia, Emmanouil Samanis, Joshua Perrett, and Awais Rashid. 2020. Catch Me If You Can: An In-Depth Study of CVE Discovery Time and Inconsistencies for Managing Risks in Critical Infrastructures. In *Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy*. 49–60. doi:10.1145/3411498.3419970
- [79] Christine Utz, Sabrina Amft, Martin Degeling, Thorsten Holz, Sascha Fahl, and Florian Schaub. 2023. Privacy Rarely Considered: Exploring Considerations in the Adoption of Third-Party Services by Websites. *Proceedings on Privacy Enhancing Technologies* 1 (2023), 5–28.
- [80] Weizhe Wang, Wei Ma, Qiang Hu, Yao Zhang, Jianfei Sun, Bin Wu, Yang Liu, Guangquan Xu, and Lingxiao Jiang. 2025. VulnRepairEval: An Exploit-Based Evaluation Framework for Assessing Large Language Model Vulnerability Repair Capabilities. arXiv:2509.03331 [cs.SE] <https://arxiv.org/abs/2509.03331>
- [81] Ying Wei, Lili Bo, Xiaobing Sun, Bin Li, Tao Zhang, and Chuanqi Tao. 2023. Automated event extraction of CVE descriptions. *Information and Software Technology* 158 (2023), 107178. doi:10.1016/j.infsof.2023.107178
- [82] Dominik Wermke, Noah Wöhler, Jan H. Klemmer, Marcel Fourné, Yasemin Acar, and Sascha Fahl. 2022. Committed to Trust: A Qualitative Study on Security & Trust in Open Source Software Projects. In *2022 IEEE Symposium on Security and Privacy (SP)*. 1880–1896. doi:10.1109/SP46214.2022.9833686
- [83] Chien-Ho Wu. 2007. An Empirical Study on the Transformation of Likert-scale Data to Numerical Scores. In *Applied Mathematical Sciences*, Vol. 1. <https://api.semanticscholar.org/CorpusID:11059810>
- [84] Meng-Jia Wu, Kelly Zhao, and Francisca Fils-Aime. 2022. Response rates of online surveys in published research: A meta-analysis. *Computers in Human Behavior Reports* 7 (2022), 100206. doi:10.1016/j.chbr.2022.100206
- [85] Qiushi Wu, Yue Xiao, Xiaojing Liao, and Kangjie Lu. 2022. OS-Aware Vulnerability Prioritization via Differential Severity Analysis. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 395–412. <https://www.usenix.org/conference/usenixsecurity22/presentation/wu-qiushi>
- [86] Susheng Wu, Wenyang Song, Kaifeng Huang, Bihuan Chen, and Xin Peng. 2024. Identifying Affected Libraries and Their Ecosystems for Open Source Software Vulnerabilities. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) (ICSE '24). Association for Computing Machinery, New York, NY, USA, Article 162, 12 pages. doi:10.1145/3597503.3639582
- [87] Yulun Wu, Zeliang Yu, Ming Wen, Qiang Li, Deqing Zou, and Hai Jin. 2023. Understanding the Threats of Upstream Vulnerabilities to Downstream Projects in the Maven Ecosystem. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1046–1058. doi:10.1109/ICSE48619.2023.00095
- [88] Corey Yang-Smith and Ahmad Abdellatif. 2025. Tracing Vulnerabilities in Maven: A Study of CVE lifecycles and Dependency Networks. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. 349–353. doi:10.1109/MSR66628.2025.00064
- [89] Siqi Zhang, Minjie Cai, Mengyuan Zhang, Lianying Zhao, and Xavier de Carné de Carnavalet. 2023. The Flaw Within: Identifying CVSS Score Discrepancies in the NVD. In *2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 185–192. doi:10.1109/CloudCom59040.2023.00039
- [90] Yuntong Zhang, Jiawei Wang, Dominic Berzin, Martin Mirchev, Dongge Liu, Abhishek Arya, Oliver Chang, and Abhik Roychoudhury. 2024. Fixing Security Vulnerabilities with AI in OSS-Fuzz. arXiv:2411.03346 [cs.CR]