Augmenting Software Bills of Materials with Software Vulnerability Description: A Preliminary Study on GitHub

Davide Fucci
davide.fucci@bth.se
Blekinge Institute of Technology
Karlskrona, Sweden

Massimiliano Di Penta
University of Sannio
Benevento, Italy
dipenta@unisannio.it

Simone Romano
siromano@unisa.it
University of Salerno
Fisciano, Italy

Giuseppe Scanniello
gscanniello@unisa.it
University of Salerno
Fisciano, Italy

Abstract

Software Bills of Material (*SBOMs*) are becoming a consolidated—and often enforced by governmental regulations—way to describe software composition. However, based on recent studies, SBOMs suffer from limited support for their consumption and lack information beyond simple dependencies, especially regarding software vulnerabilities. This paper reports the results of a preliminary study in which we augmented SBOMs of 40 open-source projects with information about *Common Vulnerabilities and Exposures* (*CVE*) exposed by project dependencies. Our augmented SBOMs have been evaluated by submitting pull requests and by asking project owners to answer a survey. Although, in most cases, augmented SBOMs were not directly accepted because owners required a continuous SBOM update, the received feedback shows the usefulness of the suggested SBOM augmentation.

CCS Concepts

• **Software and its engineering** → **Software libraries and repositories**; • **Security and privacy** → **Software and application security**.

Keywords

SBOM, VEX, Vulnerabilities management, Software repositories

ACM Reference Format:
Davide Fucci, Massimiliano Di Penta, Simone Romano, and Giuseppe Scanniello. 2025. Augmenting Software Bills of Materials with Software Vulnerability Description: A Preliminary Study on GitHub. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3696630.3728513

1 Introduction

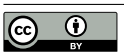
The transparency of software products in terms of their components, known as *Software Composition Analysis* (*SCA*) [3, 17], is desirable for multiple reasons. On the one side, this allows software consumers to know whether the software contains components originating from undesired providers. For example, a government might avoid using, for their applications, software components originating from hostile countries. Second, the analysis of software components helps ensure the software is not violating licensing—*e.g.*, by containing components with an incompatible license or copyrights. Last, and often more importantly, software composition allows knowing whether a piece of software depends on a vulnerable component.

A pretty consolidated and machine-readable way to describe software composition and, in general, detailing different pieces of information concerning supply-chain relationships is a *Software Bills of Material* (*SBOM*). SBOM adoption is, in recent years, not only motivated by the aforementioned needs but also by governmental regulations, such as the 2021 United States Executive Order 14028 [9] or the more recent European Union *Cyber Resilience Act* (*CRA*) [14].

Several researchers have empirically investigated the needs, adoption, and challenges of utilizing SBOMs in software development and distribution [2, 10, 11, 16, 19, 22]. On the one hand, there is a general agreement on the usefulness of SBOMs, and there is an increasing availability of tool support for SBOM generation, although the latter is generally limited to analyzing and documenting package-level dependencies from manifest/lock files. On the other hand, the studies that were conducted pointed out insufficient support in terms of tools for SBOM consumption and a limited presence of information necessary for software vulnerability assessment. More specifically, while in principle existing SBOM formats allow explicitly documenting software vulnerabilities, SBOM files (or simply SBOMs, from here onwards) generated by existing tools (*e.g.*, CycloneDX tools or the GitHub SBOM generator) merely document the (directly or indirectly) used components and their versions, and a vulnerability assessment remains a task to be conducted by the SBOM consumer using other software tools. Therefore, it appears to be importance to ask the following question:

> *To what extent would it be useful to augment SBOMs with information about components' vulnerabilities?*

Vulnerability scanners can analyze SBOM files, but their results lack system-specific context regarding a vulnerable dependency—specifically, whether the vulnerability is exploitable. A standardized approach to documenting vulnerabilities *in context* is essential to enhance transparency between dependency providers and



FSE Companion '25, June 23–28, 2025, Trondheim, Norway

ACM ISBN 979-8-4007-1276-0/2025/06
https://doi.org/10.1145/3696630.3728513

their consumers. Answering the research question above could be pivotal for bridging the gap between static software transparency tools (*e.g.,* SBOMs) and dynamic vulnerability-related information, ultimately contributing to safer and more resilient software.

To provide initial evidence and clarify this doubt, this paper reports the result of a preliminary empirical study in which we augmented the SBOMs of 40 existing open-source projects hosted on GitHub and investigated—by opening Pull Requests (PRs)—whether the project owners were willing to leverage the augmented SBOMs.

The feedback provided by project contributors who participated in the study, by either engaging in PR discussions or by answering a survey we attached to the PRs, confirmed the value of augmenting SBOMs with vulnerability-related information. At the same time, they expressed the need for a continuous generation of such pieces of information, possibly through tools integrated into the project's Continuous Integration (CI) pipeline.

Data, code, and materials necessary to replicate the findings of our study are available online [6].

2 Background and Related Work

This section provides background notions and literature analysis on SBOMs and their adoptions, needs, and challenges.

An SBOM is a formal machine-readable document that contains the details and supply-chain relationships of open-source and proprietary components used in building a software product [5]. Public administrations have been pushing software manufacturers to use SBOMs in both the United States and the European Union. Indeed, in 2021, the United States Federal Government, per President Biden's Executive Order 14028, laid down that any manufacturer releasing a software product to a federal agency must provide an SBOM of the released product [9]. In 2022, the European Commission proposed the CRA [14], which entered into force in 2024. This regulation forces software manufacturers operating in the European Union to document vulnerabilities and components of their software products with the support of SBOMs.

There are currently two main SBOM standards, namely *SPDX* (*Software Product Data Exchange*) and *CycloneDX* [22], which support the baseline component information for SBOMs [5]. Although these standards allow adding vulnerability information within an SBOM, it is recommended to use a separate document linked to the SBOM, called VEX, to store vulnerability information [5]. In particular, VEX documents are used to convey the status (*i.e.,* not affected, affected, fixed, or under investigation) of vulnerabilities within the components of a software product. SBOMs augmented with VEX documents should help software manufacturers, users, and other defenders more quickly and accurately assess the risks due to vulnerable components, which are often hidden behind opaque supply chain relationships [5].

After President Biden's Executive Order 14028 [9], there has been an increasing interest in SBOMs. In 2022, the Linux Foundation [19] surveyed 412 worldwide organizations to understand SBOM's readiness. The results of this survey show gaps in familiarity with, production planning for, and consumption of SBOMs. Xia *et al.* [22] conducted 17 interviews followed by a survey with 65 respondents to understand how practitioners perceive SBOMs. One of the main findings is that improving transparency and visibility into software products is deemed the main benefit of SBOMs. Stalknaker *et al.* [16] conducted a study combining surveys and interviews with practitioners to investigate how SBOMs are adopted by five groups of stakeholders. Stalknaker *et al.* observed that the adoption of SBOMs is still low, although practitioners have been using them in a variety of use cases—*e.g.,* software licensing or security assessment. Moreover, the study pointed out some limitations of the current technology and practice for what concerns SBOM consumption, in particular, related to security assessment. Nocera *et al.* [12] analyzed the extent to which open-source projects hosted on GitHub use SBOM generation tools and publish SBOMs. The study results indicate that, while SBOM adoption is still low, it is trending upward, and SBOMs are published in only 46% of the projects using SBOM generation tools. Kloeg *et al.* [10] interviewed groups of stakeholders involved in SBOM production and consumption to delve into the reasons for low SBOM adoption. One of the main reasons is the lack of knowledge or expertise about SBOMs. Chaora *et al.* [2] analyzed the talks on SBOMs that took place in public SBOM community engagement meetings. To promote SBOM adoption, the authors foresaw regulations and a public-recognized agency that audits and validates SBOMs submitted to a common infrastructure. Finally, Nocera *et al.* [11] interviewed 10 practitioners working in the Italian software industry about SBOMs. The study results indicate that SBOM adoption is low, yet the attention of the Italian software industry to software supply chain-related challenges is high.

To summarize, previous research has pointed out the importance of SBOM usage but also pointed out current limitations related to the technology and SBOM content. Related to the goal of our work, the latter lacks vulnerability-related information.

3 Study Methodology

As stated in the introduction, the *goal* of this study is to provide initial evidence on the perceived usefulness of augmenting SBOMs with vulnerability-related information. The *context* consists of 40 pull requests from active open-source Java projects hosted on GitHub. To address this goal, we focus on a more specific Research Question (RQ):

> *What do open-source maintainers think about integrating VEX into their existing SBOMs?*

The study of this RQ instantiates our general goal into a process aimed at collecting feedback from maintainers/developers after showing them SBOMs augmented with vulnerability information in the VEX format.

Our methodology consists of the steps presented in Figure 1. We start by searching and filtering SBOMs in GitHub, leveraging the SourceGraph [15] code search engine. To that end, we used regular expressions tailored to the two most popular SBOM formats: SPDX [18] and CycloneDX [20]. These regular expressions targeted mandatory attributes (*e.g.,* the string `SPDXID` for SPDX and `bomFormat` for CycloneDX) and specific file extensions (*e.g.,* `.yaml`, `.spdx`, `.xml`, and `.json`). Furthermore, we excluded SBOM files intended as fixtures or used for testing purposes by employing regex expressions such as `*fixture*`, `example`, and `test`. Additionally, SBOM files that did not declare any dependencies were filtered out,

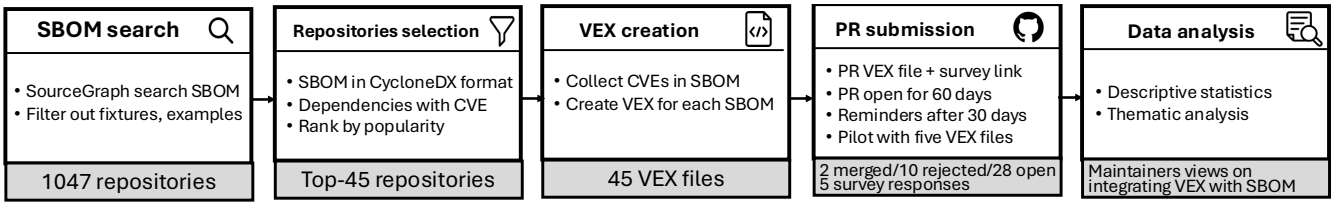


Figure 1: Methodology overview.

yielding 1,047 repositories containing relevant SBOM data. It is important to mention that the goal of this preliminary screening was to create a relatively large dataset of open-source projects hosted on GitHub and have SBOMs available in their repositories. However, in the study presented in this paper, we restricted the analysis to projects adopting the CycloneDX format only. The reason for this choice was practical and due to the adopted tool (`vexctl`) to generate VEX files, which did not support the embedded format to describe vulnerabilities required by the SPDX format. This choice reduced the number of studied repositories to 270.

Subsequently, we enriched repositories whose dependencies in the SBOM had at least one CVE according to the `osv-scanner` [8]. We selected this tool as it aggregates vulnerabilities from 25 different ecosystems (*e.g.,* `npm`, `crates.io`) and databases (*e.g.,* GitHub Security Advisory, RustSect) and provides a human and machine-readable data format to describe vulnerabilities.[1] Finally, using the GitHub API, we obtained, for each of the remaining 117 repositories, the number of stars, contributors, and total commits. The latter were used to rank repositories by popularity, from which we selected the top 45 for further analysis.

We downloaded all SBOM files present in these repositories and extracted information about vulnerabilities associated with the dependencies declared in each SBOM using `osv-scanner`. Based on the identified vulnerabilities, we generated a VEX file for each SBOM using the `vexctl` [13] tool. We selected this tool as it is one of the few existing solutions to support the generation of VEX, starting with SBOM and CVE information.

We opened a PR in each selected repository, committing the VEX file. The PR message included a link to an online survey, inviting respondents to rate their agreement on a 5-point Likert scale [4] about the perceived usefulness of using VEX to augment SBOMs with vulnerability information. Moreover, the respondents could provide qualitative feedback explaining their responses.

We left the PRs open for 60 days, during which we monitored and collected survey responses as well as comments left directly on the PRs. After 30 days of inactivity (since the PR was opened), we sent reminders to the repository maintainers. The collected data were subsequently analyzed to assess the reception and perceived usefulness of integrating VEX into SBOM workflows.

It is worth mentioning that before the actual study, we performed a pilot of two weeks targeting five randomly selected repositories. Two PRs were accepted during the pilot, while the maintainers did not review the remaining three. One maintainer filled in the survey. Since no other feedback was provided during the pilot, we decided to carry on with the study. The pilot projects were excluded from the final analysis, resulting in a total of 40 projects being considered.

[1]The vulnerability scan was run on 2024/10/03.

4 Study Results

We open a PR including the VEX file for an SBOM tracked in 40 GitHub repositories. Out of these, two were merged, 10 were rejected and 28 did not receive a review by the repository maintainers (*i.e.,* open). The survey collected five complete responses from maintainers who confirmed that they had reviewed the PR.

4.1 Analysis of Pull Request Comments

The number of comments (excluding bot-generated comments and the reminder we sent to the maintainers) for PRs that were not closed during the time of the study ranged between zero and six. For rejected PRs, the number of comments in our discussion with the maintainers ranged between one and five. Accepted PR discussion length ranged between zero and two comments.

The thematic analysis of comments regarding unmerged PRs revealed uncertainty surrounding the information contained in VEX files. Maintainers expressed willingness to adopt VEX only if provided with mechanisms for continuous (*e.g.,* triggered by SBOM generation) and automated updates (*e.g.,* integrated in CI pipelines).

Maintainers highlighted a critical limitation of the information reported in a VEX—*i.e.,* its tendency to become outdated, potentially misrepresenting the security status of project dependencies. Despite this, they recognized that the dynamic integration of VEX with SBOM could enhance overall security.

Another reason for rejecting PRs was the reliance on alternative tools, such as Dependabot [7] and Grype [1], for scanning SBOMs for vulnerabilities. However, these tools do not effectively manage the lifecycle of identified vulnerabilities by tracking their state (*e.g.,* patched or unpatched) and rationale, nor do they facilitate communication of this information to users.

An additional challenge stems from the lack of standardization across SBOM formats and the varying approaches to integrating them with VEX. This ambiguity has deterred maintainers from adopting VEX or led them to delay its integration until more stable standards emerge. One discussion cited the forthcoming European Union CRA [14] as a potential catalyst for advancing the combined use of SBOM and VEX.

Finally, while one maintainer acknowledged the importance of VEX, they emphasized its relevance primarily for internal use rather than for public projects.

4.2 Analysis of Survey Results

Among the five responses, one was from a maintainer who accepted the PR, two from maintainers who rejected it, and two from maintainers who had not yet decided after reviewing it. One respondent strongly agreed that adding vulnerability information to dependencies listed in an SBOM file is beneficial, while two partially agreed. These respondents emphasized the importance of transparency in open-source project security, noting that VEX can contribute to achieving this goal. However, they also highlighted that the effectiveness of VEX depends on the quality of the CVE databases it relies on, with its utility potentially limited by the signal-to-noise ratio in publicly available vulnerability data. The remaining two respondents disagreed, justifying their position by pointing to the availability of other tools and services capable of continuously scanning SBOM files.

4.3 Discussion

The maintainers who took part in the study expressed support for adopting VEX, provided it is automated and seamlessly integrated into project pipelines (*e.g.,* alongside SBOM creation and updates). However, some maintainers noted that VEX might be redundant since existing tools (*e.g.,* Grype) already report vulnerabilities. Nonetheless, such reports lack a standardized structure, making them non-interchangeable and unsuitable for tracking vulnerability lifecycles. More importantly, such reports cannot be easily integrated into SBOMs, the latter being considered particularly important for the studied projects (all the studied projects were adopting SBOMs already).

As a result, a key recommendation for tool providers is to support VEX format output as a *lingua franca* for sharing information about vulnerabilities in project dependencies and to integrate the VEX document generation with the SBOM generation, for example, within the project's CI pipeline.

Another significant challenge to VEX adoption is the lack of standardization. While SBOM has established two de facto standards—CycloneDX and SPDX—the former, supported by the OWASP Foundation, sponsors VEX format development, and the latter (SPDX v3.0) enables embedding vulnerability information. Such results are in line with concerns expressed in previous studies about the lack of a common standardization for SBOMs [16, 22].

Meanwhile, legislators are increasingly requiring SBOMs and vulnerability management. However, the EU CRA, which became applicable in December 2027, does not mandate a specific SBOM format or the use of VEX for vulnerability management. To align with legislative demands, further standardization of VEX is essential.

5 Threats to Validity

In the following, we discuss the threats that might affect our results based on the validity schema by Wohlin *et al.* [21].

Construct Validity. The approach we used to detect the adoption of SBOMs might affect the results. That is, the heuristic (based on the SourceGraph code search engine) to detect repositories may have missed some of them that used SBOMs. This threat is well recognized in studies similar to ours [12]. Similarly, we rely on a single tool, *vexctl*, to create VEX files based on reported CVEs.

Internal validity. A possible threat to internal validity is self-selection bias because those who engaged in the PR discussion/survey could be developers more interested in adopting the proposed SBOM augmentation. This threat is partially mitigated by, at least, already focusing on projects adopting SBOMs, and, therefore, considering supply chain documentation important.

Conclusion Validity. There could be risks that the results are affected by: (i) limited time (60 days) for contributors to review our PRs and (ii) limited activity of contributors on the repositories suggesting scant interest of developers on the hosted projects Given the preliminary nature of our study, these threats to validity are not of significant concern.

External Validity. The most important threat to this kind of validity is that our study is limited to: (i) open-source projects hosted on GitHub and (ii) SBOM generation tools owned by CycloneDX. Indeed, the second point does not represent a serious threat to validity because we opened a PR suggesting augmenting the existing SBOM with the generated VEX and asked owners to answer a survey on vulnerability-augmented SBOMs. Given our research objective, the consideration of CycloneDX (or any other SBOM standard) is not a primary concern.

6 Conclusion and Future work

In this paper, we presented the initial results of a preliminary empirical study where we enhanced the Software Bill of Materials (SBOMs) of 40 open-source projects by incorporating information about Common Vulnerabilities and Exposures (CVEs) associated with project dependencies. The augmented SBOMs have been evaluated by submitting PRs and by asking project owners to answer a survey. In most cases, our augmented SBOMs were not directly accepted because developers required a continuous SBOM update. However, the feedback of the projects' owners suggests the usefulness of the suggested SBOM augmentation.

Our preliminary results support the viability of a number of future research directions including:

- Develop and evaluate tools or methodologies for automating the continuous update of augmented SBOMs to address owners' concerns about manual maintenance.
- Investigate how to efficiently and continuously monitor dependency changes and associated CVEs.
- Expand the study to include a diverse set of open- and closed-source projects.
- Get feedback from project owners and developers to refine the presentation and usability of augmented SBOMs.

Acknowledgments

This research is partially supported by the The Knowledge Foundation of Sweden (KKS) under the SESAM project (#20230087) and S.E.R.T. Research Profile, by the European Union NEXTGenerationEU project, and by the Italian Ministry of the University and Research (MUR) Research Projects of Significant National Interest (PRIN) under MSR4SBOM project (D53D23017310001). We thank Oleksii Novikov for his contribution to the dataset creation.

References

[1] Anchore. 2020. *Grype: A vulnerability scanner for container images and filesystems.* Available at https://github.com/anchore/grype (Last Accessed: January 16, 2025).

[2] Anesu Chaora, Nathan Ensmenger, and L Jean Camp. 2023. Discourse, Challenges, and Prospects Around the Adoption and Dissemination of Software Bills of Materials (SBOMs). In *2023 IEEE International Symposium on Technology and Society (ISTAS).* IEEE, 1–4. https://doi.org/10.1109/ISTAS57930.2023.10305922

[3] Yang Chen, Andrew E. Santosa, Asankhaya Sharma, and David Lo. 2020. Automated identification of libraries from vulnerability data. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).* ACM, 90–99.

[4] Peter M Chisnall. 1993. Questionnaire design, interviewing and attitude measurement. *Journal of the Market Research Society* 35, 4 (1993), 392–393.

[5] Cybersecurity and Infrastructure Security Agency (CISA). 2024. *Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)* (3rd ed.). Available at https://www.cisa.gov/resources-tools/resources/framing-software-component-transparency-2024 (Last Accessed: January 16, 2025).

[6] Davide Fucci, Massimiliano Di Penta, Simone Romano, and Giuseppe Scanniellllo. 2025. Replication package of the paper "Augmenting Software Bills of Materials with Software Vulnerability Description: A Preliminary Study on GitHub". https://github.com/SESAM-project/SBOM-VEX-acceptance

[7] GitHub Inc. 2019. *Dependabot: Automated dependency updates built into GitHub.* Available at https://github.com/dependabot (Last Accessed: January 16, 2025).

[8] Google Inc. 2022. *OSV Scanner: Vulnerability scanner written in Go which uses the data provided by OSV.* Available at https://github.com/google/osv-scanner (Last Accessed: January 16, 2025).

[9] The United States Federal Governmen. 2021. *Executive Order on Improving the Nation's Cybersecurity | The White House.* Available at https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/ (Last Accessed: January 16, 2025).

[10] Berend Kloeg, Aaron Yi Ding, Sjoerd Pellegrom, and Yury Zhauniarovich. 2024. Charting the Path to SBOM Adoption: A Business Stakeholder-Centric Approach. In *ASIA CCS '24: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security.* ACM, 1770–1783. https://doi.org/10.1145/3634737.3637659

[11] Sabato Nocera, Massimiliano Di Penta, Rita Francese, Simone Romano, and Giuseppe Scanniello. 2024. If it's not SBOM, then what? How Italian Practitioners Manage the Software Supply Chain. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME).* IEEE, 730–740. https://doi.org/10.1109/ICSME58944.2024.00077

[12] Sabato Nocera, Simone Romano, Massimiliano Di Penta, Rita Francese, and Giuseppe Scanniello. 2023. Software Bill of Materials Adoption: A Mining Study from GitHub. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME).* IEEE, 39–49. https://doi.org/10.1109/ICSME58846.2023.00016

[13] openvex. 2023. *vexctl: A tool to make VEX work.* Available at https://github.com/openvex/vexctl (Last Accessed: January 16, 2025).

[14] The European Parliament and The Council of the European Union. 2024. *Cyber Resilience Act.* Available at https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R2847 (Last Accessed: January 16, 2025).

[15] Sourcegraph Inc. 2013. *Code Intelligence for Untangling Big, Messy Databases.* Available at https://sourcegraph.com (Last Accessed: January 16, 2025).

[16] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. 2024. BOMs Away! Inside the Minds of Stakeholders: A Comprehensive Study of Bills of Materials for Software Systems. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE).* ACM, 44:1–44:13. https://doi.org/10.1145/3597503.3623347

[17] Liran Tal. 2023. *Guide to Software Composition Analysis.* Available at https://snyk.io/series/open-source-security/software-composition-analysis-sca/ (Last Accessed: January 16, 2025).

[18] The Linux Foundation. 2011. *Software Package Data eXchange (SPDX).* Available at https://spdx.dev/use/specifications/ (Last Accessed: January 16, 2025).

[19] The Linux Foundation. 2022. *The State of Software Bill of Materials (SBOM) and Cybersecurity Readiness.* Available at https://www.linuxfoundation.org/research/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness (Last Accessed: January 16, 2025).

[20] The OWASP Foundation. 2018. *CycloneDX.* https://cyclonedx.org Available at https://cyclonedx.org(Last Accessed: January 16, 2025).

[21] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering.* Springer.

[22] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. 2023. An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE).* IEEE, 2630–2642. https://doi.org/10.1109/ICSE48619.2023.00219