BOMs Away! Inside the Minds of Stakeholders: A Comprehensive Study of Bills of Materials for Software Systems

Trevor Stalnaker
twstalnaker@wm.edu
William & Mary
Williamsburg, Virginia, USA

Nathan Wintersgill
njwintersgill@wm.edu
William & Mary
Williamsburg, Virginia, USA

Oscar Chaparro
oscarch@wm.edu
William & Mary
Williamsburg, Virginia, USA

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

Daniel M German
dmg@uvic.ca
University of Victoria
BC, Canada

Denys Poshyvanyk
denys@cs.wm.edu
William & Mary
Williamsburg, Virginia, USA

ABSTRACT

Software Bills of Materials (SBOMs) have emerged as tools to facilitate the management of software dependencies, vulnerabilities, licenses, and the supply chain. While significant effort has been devoted to increasing SBOM awareness and developing SBOM formats and tools, recent studies have shown that SBOMs are still an early technology not yet adequately adopted in practice. Expanding on previous research, this paper reports a comprehensive study that investigates the current challenges stakeholders encounter when creating and using SBOMs. The study surveyed 138 practitioners belonging to five stakeholder groups (practitioners familiar with SBOMs, members of critical open source projects, AI/ML, cyber-physical systems, and legal practitioners) using differentiated questionnaires, and interviewed 8 survey respondents to gather further insights about their experience. We identified 12 major challenges facing the creation and use of SBOMs, including those related to the SBOM content, deficiencies in SBOM tools, SBOM maintenance and verification, and domain-specific challenges. We propose and discuss 4 actionable solutions to the identified challenges and present the major avenues for future research and development.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**.

KEYWORDS

Software Bill of Materials, Survey, Interviews, Software Supply Chain, Open Source Software

ACM Reference Format:
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 '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3623347

ICSE '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3623347

1 INTRODUCTION

The software supply chain has increasingly grown in complexity with the proliferation of open-source software [76, 125] and AI/ML components [77, 78, 123]. Organizations and developers often accomplish their tasks by integrating components from a variety of vendors [69]. However, leveraging external packages does not come without a cost. The fate of a software product is intrinsically tied to its evolving dependencies [57]. If a dependency displays a vulnerability, then so too could the final product, potentially leading to severe consequences [107]. Moreover, failing to comply with the license terms of software dependencies could lead to severe legal and economic consequences for organizations [64, 115, 127, 128].

In this scenario, Software Bills of Materials (SBOMs) have emerged as mechanisms that facilitate the management of software dependencies [100], leading to improved management of software vulnerabilities, enhanced license compliance, and increased transparency in the software supply chain [99].

While SBOMs were introduced in the early 2010s [6], the 2021 US Presidential Executive Order 14028 on Improving the Nation's Cybersecurity [10] gave new momentum to SBOM formalization and adoption [9] as it required companies selling software to the US government to provide SBOMs. This was prompted by recent supply chain attacks, such as the SolarWinds breach [110] and critical vulnerabilities such as those affecting the Log4J library [87], which impacted many users [67, 97]. SBOMs are currently championed by the US National Telecommunications and Information Administration (NTIA) [98, 100] and well-known organizations such as the Linux Foundation [7] and OWASP [4]. Significant effort has been put into promoting SBOM formats and tools that can create and process SBOMs [102], with the goal of increasing adoption and fully enabling the benefits that SBOMs offer [99].

Although organizations and developers have acknowledged the importance of SBOMs and anticipate using them more frequently in the coming years [70, 120], recent research highlighted concerns regarding their commitment to SBOMs and the actual benefits SBOMs bring to their projects [70, 70, 129]. These concerns arise due to the lack of industry agreement regarding the content of SBOMs across different domains, as well as how they should be employed and integrated into their development and operational

processes [18, 70]. An additional barrier is the lack of mature tools for SBOM production and consumption [70, 129, 131].

In light of these findings, it is imperative to understand (i) how developers and other stakeholders currently create and use SBOMs, (ii) additional opportunities/benefits that SBOMs can offer for different types of software and stakeholders, (iii) the specific challenges that prevent stakeholders from fully exploiting the SBOM benefits, and (iv) actionable solutions to overcome such challenges.

This paper contributes to the body of knowledge about SBOM adoption by reporting a comprehensive empirical investigation of the aforementioned aspects. The study combined survey questionnaires with semi-structured interviews. Given the diverse types of modern software systems which SBOMs should support, we distributed five distinct questionnaires to different groups of stakeholders, resulting in a total of 150 responses (84 responses indicating SBOM familiarity). Specifically, the surveys targeted software practitioners familiar with SBOMs, contributors of critical open-source systems (OSS) [109], AI/ML, Cyber-Physical Systems (CPS) [46], and legal practitioners. To gain a deeper understanding of the key SBOM experiences, opportunities, challenges, and solutions collected in the surveys, we conducted semi-structured interviews with eight participants from different groups.

Our study expands our understanding of the limitations and challenges of SBOM formats and tools, identifies areas that research and practice on SBOM support should focus on, and provides a thorough discussion of potential solutions to overcome these barriers.

In summary, the main contributions of this paper are:

- An empirical study of SBOM adoption, challenges, and solutions. The study targeted five groups of stakeholders, according to the different types of software that SBOMs should support, offering greater scope and different perspectives about SBOM adoption compared to recent prior studies [39, 129, 131];
- A deep analysis and discussion of how software stakeholders use and create SBOMs, new opportunities/benefits that SBOM can offer, and challenges that prevent stakeholders from fully exploiting the SBOM benefits; and
- A thorough discussion and proposal of actionable solutions for the identified challenges and obstacles, as well as key areas that researchers and practitioners should focus on to improve SBOM production and consumption.

2 BACKGROUND AND RELATED WORK

Bills of Materials (BOMs) refer to the list of raw materials, components, and parts needed to manufacture an end product [95, 117]. The concept has been transferred to software systems as Software BOMs (SBOMs), which identify a project's dependencies and their provenance. Three major SBOM format specifications currently exist: SPDX [20], CycloneDX [16], and SWID [74]. While the NTIA has not officially endorsed any one specification [105], SPDX was officially recognized as a standard by ISO in 2021 [73].

Software component inventory, vulnerability analysis, and license compliance are primary SBOM use cases [62]. SPDX, supported by the Linux Foundation [7], began as a solution for managing open-source licenses and later became an SBOM standard for documenting software components, licenses, security-related information, and other metadata. CycloneDX, supported by OWASP [4], provides virtually the same features as SPDX, but focuses primarily on security and vulnerability management.

Both specifications support several file formats. SPDX: tag/value (.spdx), JSON, YAML, RDF/XML, and spreadsheets (.xls) [34]; CycloneDX: JSON, XML, and "protocol buffers" [32]. Example SBOMs for each format can be found at [55] and [25] respectively.

The differing design philosophies result in a few notable differences [62, 63]. Unlike CycloneDX, SPDX can represent code snippets within files (and their licenses), and supports annotations (adding comments to an SPDX document, *e.g.*, clarifications about ambiguous legal content). CycloneDX natively supports "compositions", which allow expressing the completeness level of a BOM element (*e.g.*, dependency relationships)—SPDX does not support this feature directly (only through annotations). CycloneDX offers more robust support for vulnerability management. For example, CycloneDX allows software suppliers to assert software vulnerabilities via the Vulnerability Exploitability eXchange (VEX) format, which SPDX does not support.

As modern software systems go beyond the mere integration of libraries and frameworks, various initiatives have proposed different types of BOMs, to account for other components typically integrated into a software system (*e.g.*, hardware devices, firmware, APIs, or AI/ML models). Practitioners have proposed BOMs for:

- external services/APIs (SaaSBOMs) [50, 54, 79];
- hardware (HBOMs) [52] and firmware (FBOMs) [35, 36, 59];
- operational (*e.g.*, configuration) environments (OBOMs) [53]; &
- datasets (DataBOMs) [42] and AI models (AIBOMs) [45, 129].

Our study targets specific populations of software stakeholders (*e.g.*, AI and Cyber-Physical Systems practitioners) to understand needs that could be potentially fulfilled by various kinds of BOMs.

While SBOMs have existed for some time [1, 6, 44, 74], they are only now beginning to be widely known [104, 130]. The analysis of their uses and shortcomings has been investigated only by a few recent studies [39, 46, 47, 70, 94, 124, 129, 131], which we discuss next.

A survey from the Linux Foundation examined the current state of SBOM usage and readiness in industry [70], aiming to identify the main use cases, benefits, and unmet needs for SBOMs. The study examined SBOM adoption, claiming that of 400 organizations surveyed worldwide, an estimated 78% would use SBOMs by 2022 and 88% by 2023. Our work differs in that we seek to identify the SBOM usage needs of developers, not organizations.

Caven *et al.* surveyed US Department of Defense officials to examine what features they look for when making procurement decisions, including features that are part of SBOMs [47]. They found that, generally, source code used in development was the least-important feature to include in SBOMs, and SBOMs were valued differently by people of different roles. In contrast to their survey, our study investigates the adoption of SBOMs from different perspectives, by targeting different sub-populations of stakeholders via distinct questionnaires and follow-up interviews.

In a study on C/C++ libraries [124], Tang *et al.* found little evidence of SBOMs being used in open-source projects. Of 24K+ GitHub repositories examined, fewer than ten contained recognizable SBOMs, yet some use package manager files, which provide similar information as SBOMs (*i.e.*, they are "quasi-SBOMs").

The gray literature review by Zahan *et al.* examined common benefits and challenges of adopting SBOMs [131]. Benefits

Table 1: Methodology and scope of SBOM studies

Study	Research methods	Considered BOMs	Study participants
Boms Away	Five surveys and follow-up interviews	SBOMs, HBOMs, AIBOMs, & DataBOMs	SBOM Producers, Consumers, Tool Makers, Standard Makers, and Educators; Developers of Critical OSS projects; AI/ML, CPS, & Legal practitioners/researchers
Xia *et al.*'s [129]	Interviews to derive one survey	SBOMs & AIBOMs	Developers
Zahan *et al.*'s [132]	Grey literature review	SBOMs	-
Linux Found. [70]	One survey	SBOMs	Software organizations

include enhanced dependency, vulnerability, risk, and licensing management, and better competitive advantage. Challenges include the lack of SBOM tooling, interoperability, and value, as well as extra effort and disclosure of sensitive information. In our work, we study how these benefits and challenges are perceived by different stakeholder groups, who use a variety of software and BOM types.

Xia *et al.* interviewed 17 software practitioners to derive 25 statements about SBOM practices, tools, & concerns. They surveyed 65 practitioners, who indicated their agreement with the statements and commented on their experience with SBOMs. Ten findings were derived from their responses, including the need to integrate SBOM formats to support various usage scenarios (*e.g.*, including vulnerability data), limited level of SBOM awareness, immaturity of SBOM tools, and lack of suitable trust mechanisms. Our study extends this prior work as it: (1) includes five surveys that target diverse stakeholders (*e.g.*, AI/ML, CPS, and legal practitioners), (2) investigates usage, challenges, & solutions for different BOMs and software types, (3) analyzes the specific challenges of creating and using SBOMs, and (4) discusses solutions to overcome these challenges.

Lin *et al.* explored the use of SBOM tools for DevSecOps and software composition analysis [93]. Balliu *et al.* compared six state-of-the-art tools that generate SBOMs for Java systems and compared how accurate the SBOMs are in listing project dependencies, compared to those given by Maven [39]. The tools capture a different set of project dependencies, missing much of the Maven dependency tree. While Balliu *et al.* discuss open challenges for accurate/effective SBOM generation and usage, our study provides a more comprehensive view of different stakeholders' problems regarding SBOMs, beyond Java systems, SBOM tools, and security-related applications.

There are different proposals to track datasets and AI model information [43, 66, 72, 96]. None of them apply the concept of BOMs to data/model supply chains. The term DataBOM was introduced and discussed by Barclay *et al.* [42] without, however, surveying developers to investigate its feasibility. Potential use cases for the AI/ML domain are mentioned, but DataBOMs are never considered within the context of AIBOMs. In our study, we ask stakeholders about the potential relationship between DataBOMs and AIBOMs.

The concept of AIBOM was proposed by Chan in 2017 [45], but no specific implementation details or recommendations were given. Barclay *et al.*, building on their previous work, explored how SBOMs might be applied in the context of AI/ML systems [41].

Table 1 provides a comparison between our study and the most related prior studies, regarding methodology & scope. A more detailed comparison can be found in our replication package [122].

3 STUDY DESIGN

The *goal* of our study is to investigate the challenges encountered by stakeholders when creating and using SBOMs, and how such challenges can be addressed. The *context* of the study consists of five stakeholder groups: software developers, project leaders and contributors, AI/ML, CPS, and legal practitioners.

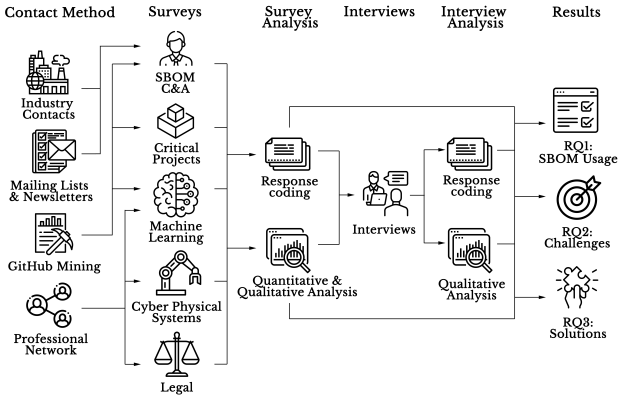

Figure 1: Research methodology (image credits at [122])

The study aims to address the following research questions (RQs):

RQ$_1$: *How do software stakeholders create and use SBOMs?*

RQ$_2$: *What are the challenges of creating and using SBOMs?*

RQ$_3$: *What are actionable solutions to SBOM challenges?*

We next describe the study methodology to answer the RQs, which includes five distinct surveys and follow-up interviews with participants from different stakeholder groups (fig. 1).

As the study involves human subjects, the methodology (including procedures to gather contact information, recruitment methods, survey/interview questions and format, data analysis, and dissemination methods) has been approved by the ethical board of the University directly involved in running the study.

3.1 Survey Design

Considering the study goal and the RQs, we have designed the survey questionnaires considering previous literature on SBOMs described in Section 2, general guidelines for survey design [68], as well as SE specific guidelines [81–85, 111].

Since the study foresees the involvement of a general population of: (1) software developers and other stakeholders that have interacted with SBOMs, and (2) domain specialists (AI/ML, CPSs, and legal practitioners), we designed questionnaires with questions asked to all stakeholder groups and questions asked to specific groups.

Table 2 summarizes the information we asked for in the surveys. A detailed description of all questions can be found in our replication package [122]. The surveys contain a mix of (five-point) Likert-scale, multiple-option, and open-ended questions that asked about: SBOM content, use cases, benefits, distribution preferences, challenges, potential solutions, dependency management practices, and legal aspects. All questionnaires also featured a consent form, a statement about data confidentiality, and a demographics section (asking about professional role, software domains, education, known programming languages, and knowledge about software security and licensing). Participants who completed the survey entered into a lottery to win one of ten $50 USD Amazon gift cards.

Table 2: Survey questions for different participant groups

Survey Group	Question Topics
SBOM Community and Adopters	SBOM content and use cases, SBOM benefits/challenges, SBOM usage for security (by role: consumers, producers, *etc.*)
Contributors of Critical OSS	SBOM content and use cases, OBOM content and adoption, other BOM practices
AI/ML Developers/Researchers	AIBOM content and use cases, DataBOM content and use cases, benefits/challenges of BOMs for AI/ML
CPS Developers/Researchers	SBOM content and use cases, HBOM content and adoption, benefits/challenges of BOMs for CPSs
Legal Practitioners	SBOM requirements, SBOMs in legal agreements, software licensing, DataBOM use cases

3.2 Participant Identification

To explore different facets of SBOMs and their usage, we identified five participant groups: SBOM Community and Adopters, contributors of critical OSS [109], as well as AI/ML, Cyber-Physical Systems (CPS), and legal practitioners.

SBOM Community and Adopters (SBOM C&A). These are people who work with SBOMs in different manners [37, 46]. Contacting people who directly use SBOMs and related technologies allowed us to obtain firsthand feedback on how SBOMs are currently used, as well as any perceived deficiencies in current SBOM standards and tools. Within this group, we identified five sub-groups of stakeholders. While we did not explicitly categorize individual stakeholders when selecting potential participants, we asked the participants to self-identify as belonging to one or more of the following groups:

- **SBOM Consumers:** People who read an existing SBOM to gather information about dependencies, vulnerabilities, or licenses.
- **SBOM Producers:** People who document a software system and its dependencies in an SBOM using a particular format (*e.g.*, SPDX, CycloneDX, or SWID).
- **SBOM Tool Makers:** People who contribute to the development of tools that facilitate the creation or use of SBOMs, *e.g.*, SBOM generators from project build scripts or dependencies.
- **SBOM Educators:** People who create or compile educational resources about SBOMs, including guides and tutorials.
- **SBOM Standard Makers:** People who contribute to specifications for the creation and usage of SBOMs. These individuals may come from government agencies, corporations, or academia.

Eligible participants for this group have been identified based on their potential experience with SBOMs, the supply chain, and software development, via a combination of three different approaches:

(1) Keyword-based search of GitHub repositories. Combining manual effort and automated tools (based on GitHub APIs [14]), we located public GitHub repositories by searching issues, commits, and files for keywords and traces related to SBOMs and the supply chain. We identified contributors who may have worked with SBOMs by locating repositories with SBOM-related files (*e.g.*, associated with the SPDX, CycloneDX, and SWID formats). From these repositories, we mined relevant commits, matching keywords such as "SBOM," "SPDX," and "bill of materials". From the matched commits, we gathered only publicly available contact information. A similar approach to identify participants was used by Xia *et al.* [129].

(2) Identifying dependencies between GitHub repositories. We found extra eligible participants by (i) examining GitHub profiles/organizations that listed projects with SBOM-related tags as topics, and (2) using GitHub's dependency feature [21] to locate dependent projects with SBOM-related tags. These repositories and their contributors logically represent groups currently using SBOMs. In total, we identified 4,423 developer email addresses via GitHub mining.

(3) Sharing the survey in relevant mailing lists. To locate additional individuals familiar with SBOMs, we published a call for participants through SBOM-related mailing lists, including the SPDX [31] and the OpenChain mailing lists [3].

Developers of Critical Open Source Systems. The Open Source Software Foundation's workgroup on Securing Critical Projects compiled a list of the 102 most critical OSS, comprising 564 total repositories [109]. The projects include the Linux Kernel, programming languages, package managers, build systems, databases, *etc.* Given the role of SBOMs in the software supply chain, we sought to administer a targeted survey examining these critical projects, which are widely depended on and may have a greater need to produce, use, and distribute SBOMs. The actions of these projects are also likely to represent and set the tone for the rest of the open-source landscape. Also, examining these projects allowed us to assess how SBOMs have spread beyond early adopters.

Using the GitHub API, we mined the top-10 contributors (by # of commits) for each of these 564 repositories. Where there were fewer than ten total contributors, we examined all that were available.

CPS Developers and Researchers. These are people with expertise in cyber-physical systems (autonomous vehicles, medical monitoring and industrial control systems, robots *etc.*), which entail a close interaction between hardware and software. Given these systems have their own supply chains and are becoming more popular in certain domains, surveying this group allowed us to examine unique challenges facing the usage of SBOMs and HBOMs, as well as how the two may interact. CPS participants were identified from our professional network.

AI/ML Developers and Researchers. These are: (i) Top-10 (by number of commits) developers that contribute to a machine learning project hosted on GitHub (with 100+ stars) and expose a public profile. AI/ML projects were identified by matching the projects' topics to keywords such as "machine learning" or "artificial intelligence" (see the full list of keywords in our replication package [122]); and (ii) AI/ML practitioners in our academic/professional network.

AI/ML components have their own supply chains, but are also increasingly integrated into traditional software products. Model/data provenance is essential to security (*e.g.*, model poisoning), licensing, usage, and research of AI/ML systems. The needs, challenges, and use cases facing AI/ML developers may be similar and different from those of typical SBOM users. By surveying this group, we aimed to understand these similarities and differences.

Legal Practitioners. Through our professional network, we identified a legal practitioner with a technical background who could answer questions about non-technical challenges facing SBOM use. This includes examining how SBOMs interact with regulations, contractual obligations, and more. The views of one respondent are not representative of the field at large, but with only a small

pool of legal practitioners having software development and SBOM experience, this group is the hardest group to survey at scale.

3.3 Survey Response Collection and Analysis

Survey responses were collected using Qualtrics [28]. Survey participants were only presented with questions related to the group(s) they selected. The survey for SBOM community and adopters was kept open for four months, with three waves of invitations. The remaining surveys were kept open for two to four weeks.

Via email and mailing list posts, we invited 4.4k+ individuals to participate in the surveys and received 229 complete responses in total (see Table 3). After removing personal information, the responses were analyzed following the procedure described below, resulting in 150 valid responses. Table 4 overviews the demographics for all the study participants.

For the closed-ended questions, we aggregated results using descriptive statistics and discussed them. In particular, we examined responses from Likert-scale questions to determine practitioner sentiments, as well as frequently-selected answers to multiple-choice questions to identify common SBOM use cases and challenges. We report the most frequently selected answers in Section 4.

For the open-ended questions, a coding approach was applied in line with [121]. Two authors ("annotators" in the following) performed a first phase of *open coding* on the first 28 valid responses of 101 received for the SBOM community and adopters survey. They independently assigned one or more codes to each response.

Once both annotators completed the open coding for the first 28 valid responses, they convened to settle disagreements and consolidated a set of labels. Since multiple codes could be assigned to each response and disagreements were discussed, we did not base our analysis on inter-rater agreements.

From this point, the remaining responses were coded by the annotators independently. During the further coding, the annotators started from the previously-established codes (available in a shared spreadsheet); yet, they had the option of adding new codes, that would, in turn, become available to the other annotator.

After the coding was completed, annotators met to discuss their coding and reconcile the disagreement cases. Results were analyzed by leveraging descriptive statistics on the codes the annotators assigned to each question. Our replication package contains the code catalog derived from the analysis for each survey and question, which includes the tag and a brief description of the code [122].

Throughout the whole coding process, the annotators flagged and reviewed answers that were nonsensical, did not answer the survey questions, were copy-pasted from the web, or appeared to be generated through ChatGPT [108]. These were reviewed by (1) inspection and discussion between annotators; (2) searching the response text using Google and validating if the text was found verbatim on the Web; or (3) validating the presence of prose, abnormal wordiness, and unusual markup characteristic of ChatGPT responses. In this way, 41 responses were removed from the analysis. Another 20 responses were removed because of numerous blank or repeated answers, and 18 were discarded as spam (*e.g.*, same email/IP addresses or identical responses). The annotators examined the survey responses and independently flagged potentially invalid responses. They discussed the cases and reached a consensus on the responses to remove and the main reason for removal.

Table 3: Number of Survey Respondents

Survey	Full Resps	Valid Resps	Fam. w/ SBOMs	Interviews
SBOM C&A	179	101	61	4
Critical	22	22	13	1
ML	21	20	8	1
CPS	6	6	1	1
Legal	1	1	1	1
Total	**229**	**150**	**84**	**8**

Role	#
P	34
C	31
TM	24
E	14
SM	16
O	7

P=Producer, C=Consumer, TM=Tool Maker, E=Educator, SM=Std. Maker, O=Other

Table 4: Abbreviated participant demographics

Survey	Top Software Domains	Top Roles	Experience (yrs)
SBOM C&A	Web apps 76% (38) Desktop apps 56% (28) Middleware 52% (26)	Programmer 30% (18) Project Lead 15% (9) Consultant 11% (7)	0-5 16% (8) 6 - 20 47% (23) 21+ 37% (18)
Critical	Web apps 68% (15) Desktop apps 45% (10) Middleware 36% (8)	Programmer 41% (9) Project Lead 27% (6) Consultant 9% (2)	0-5 5% (1) 6 - 20 45% (10) 21+ 50% (11)
ML	Deep learning 65% (13) Non-deep learning 5% (1) Both 30% (6)	ML/DL Engineer 20% (4) Researcher 20% (4) Data Scientist 15% (3)	0-5 45% (9) 6-10 50% (10) 11-15 5% (1)
CPS	-	Project Lead 17% (1) Researcher 17% (1) Programmer 17% (1)	10-15 17% (1) 16-20 67% (4) 21+ 17% (1)
Legal	-	-	13

3.4 Interviews Design and Response Analysis

We conducted one-hour semi-structured interviews with eight participants of the surveys (see Table 3), to gather deeper knowledge about their experience and responses.

Upon agreeing to answer surveys, respondents indicated willingness to be contacted for follow-up interviews. We selected respondents from the 5 surveys whose responses warranted further investigation. In particular, we sought interviews with respondents who (1) gave detailed replies highlighting interesting use cases, challenges, and potential solutions; (2) demonstrated experience in their field; and (3) diversified our interviewee pool in terms of their role (consumers, producers, *etc.*). We hoped to capture a variety of perspectives from respondents familiar with SBOMs and those that were not but had interesting thoughts on how SBOMs might affect them. Potential interviewees were identified independently by researchers during the open-coding process, and 11 participants were contacted upon consensus. In total, 8 participants accepted and completed an interview (Table 3).

The interviews were conducted in two parts. The first part asked follow-up and clarification questions which varied depending on the survey responses of each interviewee (*e.g.*, *You highlight the importance of identifiers for each software element. Why are these identifiers so important?*). For interviewees in the SBOM community and adopters group, a second part of the interview featured five questions that were common across all interviews in that group. They asked about general themes and trends observed in the survey which had a broad impact on stakeholders. Our replication package contains the protocol we followed for the interviews [122].

Interviews were conducted over Zoom and recorded with the participants' permission. The recordings were transcribed using the Whisper speech recognition tool [113]. The interviews included two authors, taking notes about the given responses. The same authors parsed and analyzed participant responses and notes individually,

employing an open coding strategy like that used in the analysis of the survey responses and discussing the coding when needed.

Interviewees were given a $50 USD Amazon gift card.

4 STUDY RESULTS

56% (84/150) of the study participants are familiar with SBOMs (see Table 3). The 22 respondents from the "Critical" survey belong to 16 of the 102 (15.7%) critical OSS.

4.1 RQ_1: SBOM Creation and Usage

4.1.1 SBOM awareness and formats. Of the 50 producers, consumers, and tool makers surveyed, 16 reported using SPDX, 8 CycloneDX, and 12 both. SWID [100] was used by only 5 respondents, often with other formats. Those that consume SBOM, do so frequently: 35.5% (11/31) of participants stated they use them daily and 29% (9/31) weekly. Of the 22 critical OSS survey participants, 9 were unfamiliar with SBOMs and 7 were aware of SBOMs, while not adopting them yet. One interviewee mentioned how the limited interest is also due to the limited tool support and the need for manually maintaining SBOMs (in line with Zahan *et al.*'s findings [131]).

It is possible that private organizations and closed-source projects use SBOMs—in any of the standard formats or their own—yet our study did not find any evidence of that. For example, it is known that CERN uses CycloneDX [75, 112] and popular standards have been mentioned by Eggers *et al.* for the nuclear industry [58].

Of 6 CPS respondents, 3 were familiar with HBOMs and 2 had used them, but with bespoke formats.

No ML practitioners surveyed were aware of BOM formats for AI systems or datasets, but one interviewed standard maker was on an SPDX team that worked on adding fields to SPDX 2.x for ML systems: fields for "describing data, the data sources, the data owners who you receive the data from, like did you buy it? Did you get it from open source? What were the references for the data you used to train the model? If it's available, also the pointer to the public information about the data". At the time of writing this paper, we have also learned that CycloneDX has added a Machine Learning Bill of Materials (ML-BOM) to its specification [51].

Participants expressed that pressure to maintain SBOMs primarily targets industry and projects at the end of a supply chain, while projects near the beginning have little incentive to produce them. Some projects, such as the Linux kernel, may have no real dependencies of their own and so do not require dependency management methods. As one interviewee noted, "I don't see a rush to add SBOMs to the originating open source. I see a rush to add SBOMs to the middle folks..."

This results in downstream components creating SBOMs on behalf of their dependencies. Other than being a cumbersome task done for somebody else; as one interviewee said, "[the risk is] miss[ing] something because you got to go back and dig back through all these different dependencies."

4.1.2 SBOM use cases, benefits, and data fields. In line with existing SBOM documentation [99] and prior studies [70, 129, 131], we found that security, dependency tracking [116], and licensing are the main use cases for SBOMs. Out of 61 SBOM practitioners, 55 mentioned as main use case dependency management, 22 licensing concerns, and 22 software security (*e.g.*, vulnerability) management. Other responses include software versioning (14), provenance (10), documentation (6), and transparency (4).

While tracking vulnerabilities was a main use case for consumers (80.7%), producers (100%), and tool makers (83.3%), some respondents were concerned that SBOMs might provide a road map of vulnerabilities for attackers. This misconception, also identified by Zahan *et al.* [131], has been addressed by NTIA [101] and our interviewees rejected the notion of "security by obscurity."

When 41 SBOM producers, tool makers, and standard makers were asked which data fields should be included in SBOMs, responses varied. The most common answers were general information about the software components: version number (24 of 41), license (22), component name (18), and a URL to the component (18). Notably, 13 respondents indicated that the SBOM should contain unique identifiers for the software component the SBOM is documenting and/or its dependencies [2, 5, 12, 13, 19, 71].

Although we found little evidence to suggest AI and DataBOMs are being used in practice, respondents mentioned two potential use cases. These BOMs could facilitate ML model reproducibility and help to identify / verify datasets across academic papers. Specifically, AIBOMs can provide transparency into how a model was trained, providing information about its architecture, hyper-parameters, and any pre-trained base models used. By providing provenance and usage information, a linked DataBOM can also make developers aware if a model was trained using a poisoned, biased, or illegally sourced dataset.

When asked about the ideal relationship between AI and DataBOMs, 9 of 20 (45%) respondents stated they should be separate documents and 5 (25%) that they should be complementary. Only 2 (10%) proposed that the documents should be combined.

The surveyed and interviewed CPS practitioners mentioned that BOMs could serve as regulatory documents for critical embedded systems (consistent with the findings of [46]), and that they could increase the transparency and reproducibility of research results in academic communities. For these tasks, the BOMs must communicate information related to the physical hardware components (part numbers, manufacturer, *etc.*), firmware, and other software (including configurations) of the system.

4.1.3 SBOM generation process, tooling, and distribution. There was little consistency in the tools used across participants, with there being a mix of in-house, commercial (*e.g.*, Anchore [22]), and open source solutions (*e.g.*, ScanCode [29]).

Despite the NTIA recommendations [103], there is currently no agreed-upon method for distributing SBOMs. Respondents have the expectation that the developers of third-party components they use should be the ones creating, maintaining, and distributing SBOMs along with their software. 5 of 12 (41.67%) critical OSS developers asked about SBOM deficiencies mentioned distribution as a challenge moving forward.

Concerning support for DataBOMs and AIBOMs, two survey participants mentioned that Hugging Face dataset cards [61] could serve as DataBOMs. Three respondents mentioned the same service's model cards, providing similar information to AIBOMs. Other tools mentioned include DVC [24] and ML-Flow [27]. These formats are "quasi-AIBOMs" since, to our knowledge, no formal AIBOM standards have been implemented and accepted in practice.

When asked when SBOMs should be generated, producers said: during each build (28/34), when publishing a major release (21/34), during deployment (19/34), and at the developer's discretion (7/34).

4.2 RQ_2: SBOM Challenges

We summarize and discuss the challenges of using and creating SBOMs, expressed by the participants.

(C1) Complexity of SBOM specifications. A common key concern among participants is the complexity of SBOM specifications, as stated in this comment: "[...] one core issue [...] is definitely a tension between use case coverage and the complexity of the spec."

Adding support for new use cases lengthens and complicates SBOM specifications. A standard maker mentioned: "[They say,] 'the spec's too complicated. All I want to do is X. [...] You're missing something for X, so I want to add that in,' which makes it more complicated for the other 99 people [without that use case]."

We noticed that the user's perception of the SBOM specification is in part determined by their use case. "If all you're interested in is licensing, [...] [you] don't want to have to learn [about other domains like security] just to be able to use the spec." However, "even if [SBOM producers] don't have that use case in mind, [their] consumers [might]."

Participants also mentioned the lack of adequate educational resources about the SBOM specifications to better communicate their content. One interviewee mentioned: "It's not just simplicity in the spec. It's not simplicity in the tooling, but how we message it and how we communicate it. Because if we send them to the [standard] spec website, they'll take a look at that and go, well, I'm not going through all that work".

(C2) Determining data fields to include in SBOMs. While some fields (software versions, licenses, or component names) are commonly agreed upon, others depend on the use case. For example, practitioners seeking to analyze their software for vulnerabilities may require BOMs to link to an external vulnerability database.

Interesting is the case of BOMs for AI/ML. AI/ML respondents expressed the need to include provenance information about datasets and models in SBOMs, to enable model verification and reproducibility. Other than standard SBOM fields, the 20 respondents from this group pointed out fields such as descriptions of the training data (17) and validation/testing data (14), preprocessing steps taken on the data (13), dataset version (13), and used optimizers/loss functions (13). When asked about fields needed in DataBOMs, they highlighted data sources (18), data transformations (18), preprocessing steps (17), dataset size (16), known/potential biases (14), and data collection procedures (14).

Of the 6 surveyed CPS practitioners, 3 expressed a need for hardware part numbers, 2 for testing and quality assurance data, 1 for system deployment information, 1 for manufacturer information and location (*e.g.*, company and geographical location), and 1 for known limitations about parts (*e.g.*, if they are not suitable for certain tasks due to security risks).

Adding additional fields to SBOM specifications makes the documents more useful, but as mentioned previously, also contributes to the complexity of the specification (C1).

(C3) Incompatibility between SBOM standards. Responses show that competing standards confuse developers. When consuming SBOMs, 23.33% of the SBOM practitioners stated that different

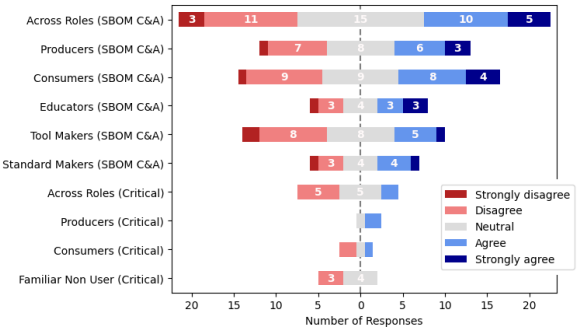

Figure 2: Perceived sufficiency of SBOM tooling.

standards pose a challenge, due to interoperability issues between standards and inconsistency between standards and tooling.

Despite this, one practitioner said: "Competition is good [...] I definitely think that we have moved faster because of CycloneDX and SPDX having this kind of competition."

There are also multiple ways of creating an SBOM for the same piece of software, often for backward compatibility reasons. One practitioner remarked: "You may have two SBOMs that technically represent the same software, but they're being produced by two different tools and they look radically different."

Fortunately, respondents suggested there are plans to increase and maintain interoperability among different standards. As one interviewee put it, "I think [the standards are] on two different paths now. [...] To say one's going to die over the other, or try to do the grand convergence and bring them together, you're just not going to, it's just going to take too long. [...] it makes much more sense to try to get the two groups to collaborate."

Addressing incompatibility between standards would likely require a community-led effort, creating clear mappings between them, and developing tools that support these mappings.

(C4) Keeping SBOMs up to date. Once an SBOM has been created, it must be maintained along with the software it represents. Substantial changes to an SBOM over time are known as SBOM drift [15]. Such changes can occur suddenly, such as a dramatic increase in the number of dependencies when an application is added to a container [80], or when new vulnerabilities are discovered in dependencies asynchronously from changes in the software — one interviewee described SBOMs as "a static vulnerability snapshot of the state of a [piece of] software at a certain point of time."

When asked about deficiencies in standards, 4.35% of participants expressed issues concerning keeping SBOM updated (1), upkeep requirements (1), and the syncing of SBOM versions (1). Of 3 critical OSS developers that consume SBOM, 1 mentioned difficulties in keeping SBOMs up-to-date. This motivates a need for tools which can dynamically update SBOMs as changes occur [114].

(C5) Insufficient SBOM tooling. Figure 2 shows stakeholders' views on whether current SBOM tools address the needs of their users. While we generally found a lack of consensus among participants, we observe that tool makers are slightly more negative. These results, combined with the participants' open-ended answers, suggest that current tool support is insufficient. One participant identified a lack of "automated ways to generate SBOM for embedded code like assembly, C, C++."

Across stakeholder groups, there was little familiarity with tools. 85% of the ML respondents were unaware of any tool support for generating AIBOMs, and 90% were unaware of tooling for DataBOMs. Only one CPS practitioner was aware of existing tools. Part of the problem may be low demand. One practitioner had used "a few [SBOM] tools [but] they [didn't] work very well," noting that "it would be nice if they were fixed" but "nobody seems to care because maybe nobody's using them."

Some projects with specific features may be unable to use current tooling, as no support exists for them yet. For example, one practitioner noted that current tooling could not "run fast on projects with tens of thousands of files... They're not designed to work with very, very large projects." Two producers faced challenges involving projects that used multiple programming languages, suggesting an unmet need for tools to support multi-language projects. Similarly, tools should be available for SBOMs to be created when only certain types of information are available, such as building SBOMs from binaries: "[T]here's source SBOMs. There's binary only SBOMs. There's SBOMs that have dependency information. There's SBOMs that have really just information about the package [...]."

(C6) Inaccurate and incomplete SBOMs. An SBOM is only as good as the information that it provides. If the information is inaccurate or incomplete, it becomes difficult for teams to make informed decisions concerning the dependencies, licensing, and security of their projects.

According to the results, currently available SBOMs are of varying quality and are often found wanting. 33% of SBOM consumers from the SBOM C&A survey mentioned poor quality SBOMs as one of the challenges they had faced in using SBOMs. 25% of the consumers from the critical OSS groups stated the same. Surprisingly, 12% of the SBOM producers had the same complaint.

Consider that the minimum SBOM requirement would be to include all direct and transitive dependency information, including the URLs of their sources. The legal practitioner we interviewed mentioned that, in his/her experience, this condition is rarely met.

Participants also discussed "false positives" in BOMs. For example, using a dependency that has a vulnerability does not necessarily mean the software will be impacted. Determining if a project is actually impacted is a more difficult problem and requires more sophisticated tooling.

The problem of inaccurate SBOMs also impacts tool developers. One respondent described how "it's been difficult to build tooling that accepts an SBOM when I'm not sure if all the fields that I'll need to depend on have been filled out."

(C7) Verifying SBOM accuracy and completeness. 33% of the critical OSS contributors mentioned how SBOM verifiability is a major challenge. This was also reported by 3 participants of the SBOM practitioner survey. That being said, the enforcement of SBOM correctness should not be so strict that it impedes SBOM creation and adoption. For example, the legal practitioner we contacted cautioned that holding BOM creators liable for inaccuracies in the documents they produce is a disincentive to creating SBOMs at all.

For security reasons, consumers will also need mechanisms to validate the integrity of an SBOM, to check that nobody has (maliciously) altered it in transit. Well-known solutions, *e.g.*, those based on hashing and checksums, can be applied to this context.

(C8) Differences across ecosystems and communities. Participants indicated that SBOM support varies across languages and package ecosystems. One interviewee mentioned: "a big part of the bottleneck is just retrieving all the information that needs to go into the SBOM and getting it from different sources [...] some language communities do a better job of capturing the metadata [to] include in the SBOM." Some respondents even suggested that tools from the same standard (*e.g.*, CycloneDX) drastically vary in quality across languages. As another participant mentioned, this "creates an ecosystem challenge for getting that data in an SBOM in a reliable way, because there are some data sources that you can't really trust."

We also observed challenges of creating SBOMs for languages with limited or no package managers. A survey respondent mentioned: "For C/C++ projects, dependencies are typically defined in autotools or cmake files, and Node, Ruby, Python, Golang, etc all have their own dependency management systems; typically recording exact versions is an output of the build process, although this doesn't come "out of the box" with C/C++ projects".

25% of the critical OSS developers surveyed who were familiar with SBOMs listed a lack of language support as a deficiency in current SBOM specifications, while 8.7% of SBOM practitioners agreed. When asked about tool deficiencies, 41.67% of critical OSS developers surveyed who were familiar with SBOMs expressed a need for more language-specific tooling.

(C9) SBOM completeness and data privacy trade-off. AI/ML participants indicated that AIBOMs and DataBOMs may entail a tradeoff between completeness and privacy on large datasets, given that these datasets may contain personally identifiable, private, sensitive, or proprietary information. CPS respondents also mentioned privacy concerns in BOMs, as CPS may actively collect and process private and sensitive data from the environment.

(C10) SBOMs for legacy packages and repositories. One interviewee expressed the challenge of generating SBOMs for legacy software, which may be deployed and used by certain user groups. Even if SBOMs become well-adopted and automatically generated during software builds, the question of what to do about legacy software remains. Software that is still regularly maintained could feasibly have an SBOM created, but it is more challenging for older systems where the original source code is missing or for systems written in languages that are now substantially less common (*e.g.*, COBOL). These languages are less likely to be supported by open-source SBOM tooling. This is particularly problematic for entities like the US government [65] or the banking industry [92]. Community-driven effort may be needed to generate, store, and share SBOMs in such situations.

An important question is, whether, for existing systems, only the newest releases require an SBOM, or if older releases that are still used by dependents also require SBOMs. The respondent said: "if ecosystems did start to publish SBOMs, [...] it would be great to see [centralized repository maintainers] go back in time, generate SBOMs for older packages"

(C11) Inability to locate dependencies for SBOMs. There may be cases where during the production or consumption of an SBOM, a certain dependency cannot be located. This could happen if a dependency was removed from a package manager (perhaps it was malicious or no longer maintained) or from the associated repository. One practitioner mentioned: "They [dependencies] may

have been yanked and removed from the upstream package registries, meaning that the mere fact of detecting that they exist could be a challenge" and "In some cases, [finding your dependency is] a lost cause in the sense that your source may be dead, the repository has disappeared and you're left to have to sift through random snapshots of archive.org calls made on the website. That's rare, but that happens." Previous work shows that malicious packages exist [38, 48, 60, 86, 88–91, 119, 132] and are commonly removed from package managers once detected [49]. Since CVEs are for vulnerabilities [11], entries for malware are not typically created, potentially leaving developers with a dead dependency reference and little way to discover the security threat the dependency poses.

A centralized database indexed on global IDs and containing provenance information for software repositories / distributions could allow developers to access critical information for projects that are no longer hosted or available. This would essentially be a third-party SBOM archive.

(**C12**) **Unclear SBOM direction and low adoption.** A recent US executive order [10] requires companies selling software to the US government to provide corresponding SBOMs. While this has created incentives to create and maintain SBOMs, our results indicate how the adoption and knowledge of SBOM are still fairly limited. Moreover, while incentives for library users are clear, those for library creators are not. Therefore, given the effort and knowledge needed for creating SBOMs, most developers forgo this effort.

The Information Technology Industry Council [18] wrote an open letter in response to recent pushes from the US federal government to mandate SBOMs [118]. They assert that SBOMs are not yet suitable contract requirements: "The presence of multiple, at times inconsistent or even contradictory, efforts suggests a lacking maturity of SBOMs." They also raise concerns about cloud services, legacy software (**C10**), and the protection of confidential or proprietary information (**C9**), all issues mentioned by respondents during our study. Though, many of these concerns have also been addressed by the NTIA [101].

This suggests a fear that the work required to create and maintain SBOMs will outweigh their benefits. As one of our practitioners said, "I hope that the hype around SBOM will lead to something that's productive [...] and will not just be something which is a compliance requirement that's going to be met in a minimal way." This fear was shared by practitioners across domains. Across our surveys, three respondents expressed worry that SBOMs would not be useful and another three feared that they would be time-consuming.

Lastly, as we were reminded by numerous respondents, SBOMs are still not-mature-yet technology that will take time to mature. Currently, there is still a need to motivate and implement support for consumer use cases. In an interview, one respondent stated, "You know, if you are a large organization and, say, you take a magic wand, and tomorrow all your software vendors start to provide accurate SBOMs, what are you going to do with this?"

4.3 RQ_3: Solutions to SBOM Challenges

In this section, we discuss solutions for the identified challenges. The proposed solutions do not address (**C3**), (**C10**), & (**C11**): these require additional research to mitigate effectively, but insights and potential directions are discussed in their challenge descriptions. Table 5 provides a summary of the proposed solutions.

(**S1**) **Multi-dimensional SBOM specifications.** We identify three dimensions that contribute to the complexity SBOM specification: (1) the intended use case of an SBOM, (2) the type of software the SBOM is generated for, and (3) the amount of information documented in an SBOM. Providing clear guidance for these dimensions is needed to inform consumers/producers which fields an SBOM should contain (**C2**). The ultimate goal is to reduce the cognitive load placed on users of the specification (**C1**).

SBOM use cases. As discussed, dozens of potential use cases exist for SBOMs [8], but including fields tailored for each of these results in cluttered specifications (see (**C1**)). In interviews, we learned that the SPDX team is working on *profiles* [106] which define the fields required in an SBOM document meant for a specific use case. This will allow producers to create SBOMs tailored to their use case without worrying about irrelevant fields to them. One practitioner mentioned that "being able to call [use case] out in these profiles will make [what to expect in the quality] a lot clearer. And I think that might help with [poor quality SBOMs] (**C6**). Not so much making the quality of the SBOMs better, but at least making it obvious what the quality is." Another said: "Let's say I want to just graph the relationships, right? There's a lot of data that's included in the SBOM that I wouldn't necessarily need. And if some of that data is expensive to calculate, then the tool that gives me the SBOM would run a lot faster if all I was ever looking for was a way to kind of graph the relationships."

Types of software. Different types of software require different information to adequately describe them. ML-related software requires fields that firmware or cloud services likely will not. Even though all three fall under the umbrella of software, it may be prudent to separate them into distinct SBOM types (AIBOM, FBOM, SaaSBOM, *etc.*), so that it is easier for end users to know the type of system the SBOM describes. This model of different SBOM types has already been adopted by CycloneDX [56].

Amount of information in SBOMs. Within the same use case and software type, users may desire different amounts of data in an SBOM. One practitioner noted: "it would be interesting to have different levels [...] where this has 'level 1' data. [...] This tool generates 'level 2' data, this tool generates 'level 3' data...". These data levels reflect the amount of information a user can expect to find in the SBOM. Lower data levels could potentially be used for privacy-sensitive applications (**C9**). Data levels could also create some standardization in tooling: "I think it would help people who are writing tools [...] to be able to then differentiate between the level of data that they can expect to see within the SBOM."

Adding this flexibility to standards does not necessarily make them more complex or difficult to use. One practitioner indicated that "even though the minimum requirements that have been provided [...] seem to be or could be construed as daunting, the essence of what needs to be provided in SBOM can be surprisingly simple." Educational resources and documentation will have to be well-crafted to explain this approach.

(**S2**) **Enhanced SBOM tooling and build system support.** Across all surveys, three respondents suggested better libraries as a tooling solution. One said, "Increased investment in open source libraries that can be incorporated in end user commercial and open

Table 5: How each SBOM solution addresses the SBOM challenges and the roles impacted by the challenges/solutions

(S1) Multi-dimensional SBOM specifications		
- Structure SBOM specifications considering three dimensions, i.e., use cases, types of software, and amount of information needed (*a.k.a.* information level) - Create more structured, easy-to-navigate, and easy-to-search specifications - Improve educational material about SBOM specifications		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C1) Complexity of SBOM specifications	Shorter and easy-to-browse SBOMs specs. without unneeded information (per use case, system, *etc.*)	P, C, TM, E, SM
(C2) Determining data fields to include in SBOMs	Optional SBOM fields added only when required. Information levels determine optional, recommended, and mandatory SBOM fields.	P, C, TM, E, SM
(C6) Inaccurate and incomplete SBOMs	SBOMs not incomplete if irrelevant/hard-to-find info for a given use case is not required.	P, C, TM
(C9) SBOM completeness and data privacy trade-off	SBOMs can tailor the required fields for data privacy according to defined information levels.	P, C, TM, SM
(S2) Enhanced SBOM tooling and build system support		
- Develop libraries and base infrastructure for SBOM production, consumption, and verification - Develop SBOM tooling for binaries and programming languages with no package managers - Integrate SBOM creation into build and continuous integration (CI) systems and AI/ML frameworks (TensorFlow, *etc.*)		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C4) Keeping SBOMs up to date	SBOM tools compatible with build and CI/CD automation to create/update SBOMs at each build.	P, C, TM
(C5) Insufficient SBOM tooling	Improved SBOM tools with support for multiple programming languages and AI/ML frameworks.	P, C, TM
(C6) Inaccurate and incomplete SBOMs	SBOM tools integrated with build automation and AI/ML frameworks create SBOMs with dependencies actually used in binaries, releases, and AI/ML models.	P, C, TM
(C7) Verifying SBOM accuracy and completeness	Tools to check that SBOMs created from source code and binaries contain the same dependency info.	P, C, TM
(C8) Differences across ecosystems and communities	Improved SBOM tools would lead to increased SBOM adoption across languages and ecosystems.	P, C, TM
(S3) Strategies for SBOM verification		
- Third-party (community-based) certification/verification of SBOMs		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C6) Inaccurate and incomplete SBOMs	With verification mechanisms in place, certified SBOMs would be more accurate and complete.	P, C, TM
(C7) Verifying SBOM accuracy and completeness	Verifying and certifying SBOM content leads to enhanced accuracy and completeness.	P, C, TM
(S4) Increasing incentives for SBOM adoption		
- Create mandates to create and use SBOMs for different stakeholders - Minimize the effort to create and maintain SBOMs (*e.g.*, by developing tools integrated with existing systems and processes) - Increase motivation to develop (open-source) SBOM tooling (e.g., via integration and badging in code repositories such as GitHub) - Promote SBOMs benefits/usage and improve educational materials (e.g., by promoting successful cases of SBOM usage and tooling)		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C5) Insufficient SBOM tooling	Increased incentives for SBOM adoption would drive further development of SBOM tooling.	P, C, TM, E, SM
(C12) Unclear SBOM direction	SBOM mandates and promotion/education materials clarify SBOM benefits and usage costs. SBOM incentives would better involve open-source communities in SBOM creation/usage/promotion.	P, C, TM, E, SM

source tools [can address current deficiencies in tooling]." Well-maintained, easy-to-use libraries would serve as the foundation and motivation to develop SBOM tools (**C5**) providing functionality for creating, maintaining (**C4**), parsing, and managing SBOMs, enhancing the user experience and, potentially, SBOM adoption.

Our findings indicate the need for language-specific SBOM production tools. A language-agnostic tool is unlikely to adequately support all scenarios. As such, there is work to be done creating SBOM generation tools for different ecosystems, including resolving disparities in the quality of available tools. Creating better tools will be a community effort: "part of it is just [...] being willing to get in and help out with the quality of those tools." Language-specific tooling can be built on language-agnostic libraries (**C8**).

SBOMs will likely become more accurate and complete with better tool support (**C6**). Respondents from the critical OSS survey pointed out that quasi-SBOM files are typically accurate and are generated/checked automatically by tools: mature SBOM tools would likely be able to perform similarly.

Moreover, in the current landscape of varying SBOM quality, consumption tools may also be responsible for checking the accuracy of the SBOMs consumed (**C7**). A respondent noted that consumption tools "have a perhaps harder job to make sure that the data that's being generated is accurate."

Furthermore, existing build systems (*e.g.*, Maven or Gradle) should be made SBOM-aware: capable of reading and generating SBOMs: "[O]ne way [for SBOMs to be easier to use] would be for build tools to start generating them without asking." We have observed from our surveys that developers tend to prefer processes or tools that are commonly used or predetermined: "when the recommended way of doing something is the default, then it gets done more often." SBOM generation functionality in build tools would more easily facilitate the update of SBOMs (**C4**).

We have seen that developers rely on package management systems to obtain a list of their project's dependencies. Many of these systems also provide quasi-SBOM files. If SBOM generation and acquisition could be handled at the package manager level, we would likely see a large uptick in adoption (**C12**). SBOMs could be stored along with other package information and queried through APIs. Indeed, interviewed practitioners suggested that SBOMs should be kept as close to the source as possible. As an SBOM moves further from the source, it is less likely to be up-to-date (**C4**).

GitHub recently unveiled new functionality capable of generating SPDX documents for a cloud repository [17]. Through integration with GitHub's Dependency Graph tool [33], this capability supports SBOM generation for a number of popular languages and is easily accessible to developers, marking a strong start for SBOM integration.

It was also suggested that ML libraries could generate AIBOMs or play an integral part in easily accessing required information: "eventually there'll be [...] something built into TensorFlow or PyTorch [...] that outputs a document [...] that tells you the key elements [like] the hyper-parameters."

(**S3**) **Strategies for SBOM verification.** One initially apparent method to approach incomplete or incorrect SBOMs would be to hold parties accountable for the SBOMs they generate (**C6**), but this could lead to unintended consequences. A legal practitioner said, "[a] requirement for them to certify that it is complete or correct is only going to result in fear of creating SBOMs. 'Perfect' should not be the enemy of 'good.'" Beyond this, SPDX SBOMs are licensed under Creative Commons 0 (CC0) [23, 30], meaning no warranty is included and the producer assumes no liability. The open-source licensing of tools protects their creators from litigation since many licenses also do not provide a warranty [26]. According to the legal practitioner we interviewed, issues of liability would likely only arise if proprietary software or service provided a warranty. He/she "could see there being contractually accepted liability as part of [one party agreeing to provide an accurate SBOM]."

Two other solutions emerged from our surveys (**C7**). A third-party certification or review board could approve SBOMs and endorse them. However, as one respondent put it, "central authorities have never seemed to work too well in our industry [...]". An alternative, decentralized approach could involve the assessment of SBOMs by their consumers and other stakeholders, with issues reported to the SBOM producer or posted in a shared database.

(**S4**) **Increasing incentives for SBOM adoption.** There is a need to either minimize the effort needed to create and maintain SBOMs (such as improving current development toolkits to generate them) or by gaining other benefits, such as having tools that consume SBOM and help with developer tasks. Similarly, it is necessary to motivate the creators of the development toolkits to support SBOM creation (**C5**). Github's new SBOM tools are a step in the right direction. Also, issuing badges might be a simple incentive that might promote the adoption of SBOMs (as it has been in other domains [126]) (**C12**).

Similar to Executive Order 14028, other stakeholders could require their participants to provide SBOMs. For example, the scientific publication of tools and models could require that artifacts be accompanied by SBOMs (**C12**). These SBOMs would increase the transparency of the work and ideally increase reproducibility.

At the same time, better marketing and educational materials that emphasize the importance of SBOMs are needed, both for software developers and consumers. As one user put it, "It's not just simplicity in the spec [nor] simplicity in the tooling, but how we message it and how we communicate it."

Ultimately, creating and using SBOMs should be done because it helps to create and maintain better, more secure, and reliable software, and that ultimately benefits society.

5 THREATS TO VALIDITY

External Validity. The conclusions of our study apply to the population that participated in the survey and interviews. By design, we cannot overly generalize our results [40], yet our observations pertaining to open-source developers may extend to other open-source projects. Generalizability for the industry is more difficult, but industries within the same country will abide by the same legislation and regulations, likely resulting in similar use cases and challenges. Ultimately, our goal was not to claim generalizability, but to gain a clearer understanding of the current landscape of SBOM usage, the challenges therein, and how to overcome them. While the number of respondents for the ML, CPS, critical, and legal surveys is rather small, they provided insights from the perspectives of practitioners (belonging to different areas) who may or may not use SBOMs firsthand, which are still valuable to understand the current landscape and future directions of SBOMs.

Internal Validity. To mitigate researcher bias in open-ended response coding, we followed an iterative, hybrid coding process that included discussion for all disagreements to reach a consensus such that the codes applied to a given response most accurately reflected its content. To ensure that we surveyed practitioners with different backgrounds, we employed a diverse set of strategies to find participants, including the search for relevant repositories on GitHub, posting to relevant mailing lists, and contacting practitioners through our professional network. However, the low response rate and self-selection bias may have influenced the results by attracting participants interested in the survey topic. We formulated our survey/interview questions to follow best practices and survey/interview design guidelines. We ensured questions were clear and concise, avoiding language that would bias respondents towards a certain answer, and providing clarification and defining terms we used when necessary. Additionally, we mitigated potential confirmation bias in our qualitative analysis by performing independent coding, discussing disagreements, and reaching a consensus backed with facts from the data. While we attempted to remove AI-generated responses from results, they remain a well-accepted risk in this kind of study.

6 CONCLUSION

This paper reports and discusses the findings from a study—conducted through surveys and interviews with software practitioners—on the use of bills of materials for software systems. Other than targeting a general population of SBOM adopters, we also targeted specialized populations of developers of critical OSS, as well as AI/ML, CPS, and legal practitioners.

The study results indicate that, while the adoption of SBOMs is still low, practitioners utilize them in a variety of use cases at various stages of software development and maintenance, including software licensing, dependency management, and security assessment. While SBOMs have the potential to aid in both research and industry, tool support and SBOM standards are nearly nonexistent in specific areas such as AI/ML and CPS.

The wide variety of use cases for SBOMs, and the complexity and heterogeneity of software systems, have led to numerous challenges, such as the complexity of standard specifications, inadequate tooling, or data privacy vs. completeness tradeoffs. To address such challenges, our study has identified a number of solutions and opened the road for future research and development in this area.

DATA AVAILABILITY

We provide an anonymized replication package containing survey and interview protocols, aggregated results, a code catalog for survey and interview responses with definitions, code to process results, and other data required for verifiability [122].

ACKNOWLEDGEMENTS

We thank the study participants for their time and valuable contributions. This research was partially funded by NSF CCF-2217733. A complete, detailed list of image attributions can be found at [122].

REFERENCES

[1] [n. d.]. CycloneDX History. https://cyclonedx.org/about/history/.
[2] [n. d.]. GitOID. https://www.iana.org/assignments/uri-schemes/prov/gitoid.
[3] [n. d.]. OpenChain Main Mail List. https://lists.openchainproject.org/g/main.
[4] [n. d.]. OWASP. https://owasp.org/.
[5] [n. d.]. Software Heritage. https://www.softwareheritage.org/.
[6] [n. d.]. SPDX Overview. https://spdx.dev/about/.
[7] [n. d.]. The Linux Foundation. https://www.linuxfoundation.org/.
[8] 2013. SPDX Technical Team Use Cases 2.0. https://wiki.spdx.org/view/Technical_Team/Use_Cases/2.0. Accessed: 2023-29-03.
[9] 2016. Cybersecurity Supply Chain Risk Management. https://csrc.nist.gov/projects/cyber-supply-chain-risk-management
[10] 2021. EXECUTIVE ORDER 14028. https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity
[11] 2021. What is a CVE? https://www.redhat.com/en/topics/security/what-is-cve.
[12] 2022. Annex F External repository identifiers (Normative). https://spdx.github.io/spdx-spec/v2.3/external-repository-identifiers/#f42-gitoid.
[13] 2022. Common Platform Enumeration (CPE). https://csrc.nist.gov/Projects/Security-Content-Automation-Protocol/Specifications/cpe.
[14] 2022. GitHub REST API documentation. https://docs.github.com/en/rest?apiVersion=2022-11-28. Accessed: 2023-28-03.
[15] 2022. SBOM Drift. https://docs.anchore.com/current/docs/sbom_management/sbom_drift/.
[16] 2023. CycloneDX Specifications. https://github.com/CycloneDX/specification
[17] 2023. Introducing self-service SBOMs. https://tinyurl.com/mt9jwcdx.
[18] 2023. ITI. https://www.itic.org/.
[19] 2023. purl-spec. https://github.com/package-url/purl-spec.
[20] 2023. SPDX Specifications. https://spdx.dev/specifications/
[21] [n.d.]. About the dependency graph. https://tinyurl.com/28r3v6e2. Accessed: 2023-28-03.
[22] [n.d.]. Anchore. https://anchore.com/platform/. Accessed: 2023-29-03.
[23] [n.d.]. CC0 1.0 Universal (CC0 1.0) Public Domain Dedication. https://creativecommons.org/publicdomain/zero/1.0/. Accessed: 2023-29-03.
[24] [n.d.]. Data Version Control. https://dvc.org/. Accessed: 2023-29-03.
[25] [n.d.]. Example of an SPDX SBOM. https://github.com/spdx/spdx-examples/blob/master/example1/spdx2.2/example1.spdx.
[26] [n.d.]. The MIT License. https://opensource.org/license/mit/.
[27] [n.d.]. mlflow. https://mlflow.org/. Accessed: 2023-29-03.
[28] [n.d.]. Qualtrics. https://www.qualtrics.com/. Accessed: 2023-28-03.
[29] [n.d.]. ScanCode. https://www.nexb.com/scancode/. Accessed: 2023-29-03.
[30] [n.d.]. SPDX Object Property: dataLicense. https://spdx.org/rdf/spdx-terms-v2.1/objectproperties/dataLicense___1140128580.html. Accessed: 2023-29-03.
[31] [n.d.]. spdx@lists.spdx.org. https://lists.spdx.org/g/spdx. Accessed: 2023-28-03.
[32] [n.d.]. Specification Overview. https://cyclonedx.org/specification/overview/.
[33] [n.d.]. Supported package ecosystems. https://docs.github.com/en/code-security/supply-chain-security/understanding-your-software-supply-chain/about-the-dependency-graph#supported-package-ecosystems.
[34] [n.d.]. Using SPDX. https://spdx.dev/resources/use/.
[35] Amy Nelson, Jiewen Yao, Vincent Zimmer. 2021. Traceable Firmware Bill of Materials Overview. https://tinyurl.com/2p8ujxau.
[36] Andrei Costin. 2022. Securing Your Iot Device With Fboms From Devastating Cyberattacks. https://euhubs4data.eu/blog/securing-iot-device-with-fboms/.
[37] Arushi Arora, Virginia Wright, and Christina Garman. 2022. Strengthening the Security of Operational Technology: Understanding Contemporary Bill of Materials. *JCIP The Journal of Critical Infrastructure Policy* 3, 1 (2022), 111.
[38] Aadesh Bagmar, Josiah Wedgwood, Dave Levin, and Jim Purtilo. 2021. I Know What You Imported Last Summer: A study of security threats in the Python ecosystem. *arXiv preprint arXiv:2102.06301* (2021).
[39] Musard Balliu, Benoit Baudry, Sofia Bobadilla, Mathias Ekstedt, Martin Monperrus, Javier Ron, Aman Sharma, Gabriel Skoglund, César Soto-Valero, and Martin Wittlinger. 2023. Challenges of Producing Software Bill Of Materials for Java. *arXiv preprint arXiv:2303.11102* (2023).
[40] Sebastian Baltes and Stephan Diehl. 2016. Worse than spam: Issues in sampling software developers. In *Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement*. 1–6.
[41] Iain Barclay, Alun Preece, Ian Taylor, Swapna Krishnakumar Radha, and Jarek Nabrzyski. 2022. Providing assurance and scrutability on shared data and machine learning models with verifiable credentials. *Concurrency and Computation: Practice and Experience* (2022), e6997.
[42] Iain Barclay, Alun Preece, Ian Taylor, and Dinesh Verma. 2019. Towards traceability in data ecosystems using a bill of materials model. *arXiv* (2019).
[43] Emily M Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics* 6 (2018), 587–604.
[44] "Bill Bensing". 2022. History of the Software Bill of Material (SBOM). https://billbensing.com/software-supply-chain/history-software-bill-of-material-sbom/.
[45] Brian Ka Chan. 2017. Artificial Intelligence Bill of Materials (AI-BOM). https://minddata.org/bill-of-artificial-intelligence-materials-boaim-Brian-Ka-Chan-AI.
[46] Seth Carmody, Andrea Coravos, Ginny Fahs, Audra Hatch, Janine Medina, Beau Woods, and Joshua Corman. 2021. Building resilient medical technology supply chains with a software bill of materials. *NPJ Digital Medicine* 4, 1 (2021), 34.
[47] Peter J Caven, Shakthidhar Reddy Gopavaram, and L Jean Camp. 2022. Integrating Human Intelligence to Bypass Information Asymmetry in Procurement Decision-Making. In *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*. IEEE, 687–692.
[48] Kai Chen, Xueqiang Wang, Yi Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Bin Ma, Aohui Wang, Yingjun Zhang, and Wei Zou. 2016. Following devil's footprints: Cross-platform analysis of potentially harmful libraries on android and ios. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 357–376.
[49] Catalin Cimpanu. 2017. Ten Malicious Libraries Found on PyPI - Python Package Index. https://www.bleepingcomputer.com/news/security/ten-malicious-libraries-found-on-pypi-python-package-index/. Accessed: 2023-27-03.
[50] Cloud Security Alliance. 2022. SaaS Governance Best Practices for Cloud Customers. https://cloudsecurityalliance.org/artifacts/saas-governance-best-practices-for-cloud-customers/.
[51] CycloneDX. [n. d.]. https://cyclonedx.org/.
[52] CycloneDX. 2022. Hardware Bill of Materials (HBOM). https://github.com/CycloneDX/bom-examples/tree/master/HBOM.
[53] CycloneDX. 2022. Operations Bill of Materials (OBOM). https://github.com/CycloneDX/bom-examples/tree/master/OBOM.
[54] CycloneDX. 2022. Software-as-a-Service BOM (SaaSBOM). https://github.com/CycloneDX/bom-examples/tree/master/SaaSBOM.
[55] CycloneDX. 2022. Software Bill of Materials (SBOM). https://github.com/CycloneDX/bom-examples/tree/master/SBOM.
[56] CycloneDX. [n.d.]. Capabilities. https://cyclonedx.org/capabilities/.
[57] Massimiliano Di Penta, Daniel M. Germán, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2010. An exploratory study of the evolution of software licensing. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*. 145–154. https://doi.org/10.1145/1806799.1806824
[58] Shannon Leigh Eggers, Drew Christensen, Tori Brooke Simon, Baleigh Rae Morgan, and Ethan S Bauer. 2022. *Towards Software Bill of Materials in the Nuclear Industry*. Technical Report. Idaho National Lab.(INL), Idaho Falls, ID (United States).
[59] Eliot Beer. 2022. Firmware security in the spotlight after novel ransomware attacks. https://thestack.technology/firmware-attacks-focus/.
[60] William Enck and Laurie Williams. 2022. Top Five Challenges in Software Supply Chain Security: Observations From 30 Industry and Government Organizations. *IEEE Security Privacy* 20, 2 (2022), 96–100. https://doi.org/10.1109/MSEC.2022.3142338
[61] Hugging Face. [n.d.]. Dataset Cards. https://huggingface.co/docs/hub/datasets-cards. Accessed: 2023-29-03.
[62] FOSSA Inc. [n. d.]. A Practical Guide to CycloneDX. https://fossa.com/learn/cyclonedx.
[63] FOSSA Inc. 2023. CycloneDX vs SPDX. https://www.youtube.com/watch?v=IQledp8WccU.
[64] GR Gangadharan, Vincenzo D'Andrea, Stefano De Paoli, and Michael Weiss. 2012. Managing license compliance in free and open source software development. *Information Systems Frontiers* 14 (2012), 143–154.
[65] GAO. 2016. Federal Agencies Need to Address Aging Legacy Systems. https://www.gao.gov/assets/files.gao.gov/assets/gao-16-696t.pdf.
[66] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.
[67] Google. 2021. Understanding the Impact of Apache Log4j Vulnerability. https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html.
[68] Robert M. Groves, Floyd J. Jr. Fowler, Mick P. Couyper, James M. Lepkowski, Eleanor Singer, and Roger Tourangeau. 2009. *Survey Methodology, 2nd edition*. Wiley.
[69] GuardRails. 2023. What is a Software Bill of Materials, and Why is it Important For Security? https://www.guardrails.io/blog/what-is-a-software-bill-of-materials-and-why-is-it-important-for-security/. Accessed: 2023-29-03.
[70] Stephen Hendrick. 2022. Software Bill of Materials (SBOM) and Cybersecurity Readiness. https://tinyurl.com/293v3xte.
[71] Henk Birkholz, Jessica Fitzgerald-McKay, Charles Schmidt, David Waltermire. 2021. Concise Software Identification Tags. https://www.ietf.org/archive/id/draft-ietf-sacm-coswid-19.html.
[72] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. 2018. The dataset nutrition label: A framework to drive higher data quality standards. *arXiv preprint arXiv:1805.03677* (2018).
[73] ISO. 2021. ISO/IEC 5962:2021 Information technology — SPDX Specification V2.2.1. https://www.iso.org/standard/81870.html.
[74] ISO. 2023. ISO/IEC 19770-2:2015. https://www.iso.org/standard/65666.html.

[75] Laman Jalilova. 2021. Consolidation Of Cern Accelerator Build Infrastructure. https://cds.cern.ch/record/2778929/files/Laman_Jalilova_CERN_Report.pdf.
[76] Andrew Jamieson. 2020. Quantifying Complexity: The Challenges of Supply Chain Security. https://www.eetimes.com/quantifying-complexity-the-challenges-of-supply-chain-security/. Accessed: March 26, 2023.
[77] Wenxin Jiang, Nicholas Synovic, Matt Hyatt, Taylor R Schorlemmer, Rohan Sethi, Yung-Hsiang Lu, George K Thiruvathukal, and James C Davis. 2023. An empirical study of pre-trained model reuse in the hugging face deep learning model registry. *arXiv preprint arXiv:2303.02552* (2023).
[78] Wenxin Jiang, Nicholas Synovic, Rohan Sethi, Aryan Indarapu, Matt Hyatt, Taylor R Schorlemmer, George K Thiruvathukal, and James C Davis. 2022. An Empirical Study of Artifacts and Security Risks in the Pre-trained Model Supply Chain. In *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. 105–114.
[79] John P. Mello Jr. 2022. SBOMs in the SaaS era: 5 reasons why you should consider a SaaSBOM. https://tinyurl.com/36pe3vvh.
[80] Josh Bressers. 2022. Fast and Furious: Doubling Down on SBOM Drift. https://thenewstack.io/fast-and-furious-doubling-down-on-sbom-drift/.
[81] Barbara A. Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research part 2: designing a survey. *ACM SIGSOFT Software Engineering Notes* 27, 1 (2002), 18–20.
[82] Barbara A. Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research: part 3: constructing a survey instrument. *ACM SIGSOFT Software Engineering Notes* 27, 2 (2002), 20–24.
[83] Barbara A. Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research part 4: questionnaire evaluation. *ACM SIGSOFT Software Engineering Notes* 27, 3 (2002), 20–23.
[84] Barbara A. Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research: part 5: populations and samples. *ACM SIGSOFT Software Engineering Notes* 27, 5 (2002), 17–20.
[85] Barbara A. Kitchenham and Shari Lawrence Pfleeger. 2003. Principles of survey research part 6: data analysis. *ACM SIGSOFT Software Engineering Notes* 28, 2 (2003), 24–27.
[86] Ravie Lakshmanan. [n. d.]. Researchers Uncover 29 Malicious PyPI Packages Targeted Developers with W4SP Stealer. https://thehackernews.com/2022/11/researchers-uncover-29-malicious-pypi.html. Accessed: 2023-27-03.
[87] Ravie Lakshmanan. 2021. Extremely Critical Log4J Vulnerability Leaves Much of the Internet at Risk. https://thehackernews.com/2021/12/extremely-critical-log4j-vulnerability.html. Accessed: 2022-05-12.
[88] Ravie Lakshmanan. 2022. Malicious NPM Package Caught Mimicking Material Tailwind CSS Package. https://thehackernews.com/2022/09/malicious-npm-package-caught-mimicking.html. Accessed: 2023-27-03.
[89] Ravie Lakshmanan. 2022. Multiple Backdoored Python Libraries Caught Stealing AWS Secrets and Keys. https://thehackernews.com/2022/06/multiple-backdoored-python-libraries.html. Accessed: 2023-27-03.
[90] Ravie Lakshmanan. 2022. Researchers Uncover PyPI Package Hiding Malicious Code Behind Image File. https://thehackernews.com/2022/11/researchers-uncover-pypi-package-hiding.html. Accessed: 2023-27-03.
[91] Genpei Liang, Xiangyu Zhou, Qingyu Wang, Yutong Du, and Cheng Huang. 2021. Malicious Packages Lurking in User-Friendly Python Package Index. In *2021 IEEE 20th International TrustCom*. IEEE, 606–613.
[92] Everist Limaj, Edward Bernroider, and Maria Ivanova. 2020. Facing Legacy Information System Modernization in Scaling Agility in the Banking Industry: Preliminary Insights on Strategies and Non-technical Barriers. (2020).
[93] Lu Lin et al. 2023. Generating Software Bill of Material for Vulnerability Management and License Compliance. (2023).
[94] Robert Alan Martin. 2020. Visibility & control: addressing supply chain challenges to trustworthy software-enabled things. In *SSS'20*. IEEE, 1–4.
[95] Jeffrey G. Miller and Linda G. Sprague. 1975. Behind the Growth in Materials Requirements Planning. https://hbr.org/1975/09/behind-the-growth-in-materials-requirements-planning. *Harvard Business Review* (1975).
[96] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*. 220–229.
[97] NIST. 2021. CVE-2021-44228. https://nvd.nist.gov/vuln/detail/CVE-2021-44228.
[98] NTIA. 2019. Framing Software Component Transparency: Establishing a Common Software Bill of Material (SBOM). https://tinyurl.com/ya978te4.
[99] NTIA. 2019. Roles and Benefits for SBOM Across the Supply Chain. https://ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf.
[100] NTIA. 2021. SBOM at a Glance. https://tinyurl.com/txyvbhfu.
[101] NTIA. 2021. SBOM Myths vs. Facts. https://tinyurl.com/57rvensd
[102] NTIA. 2021. SBOM Tool Classification Taxonomy. https://ntia.gov/files/ntia/publications/ntia_sbom_tooling_taxonomy-2021mar30.pdf.
[103] NTIA. 2021. Sharing and Exchanging SBOMs. https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf.
[104] NTIA. 2021. Software Bill of Materials Elements and Considerations. https://ntia.gov/sites/default/files/publications/uscc_-_2021.06.17_0.pdf.
[105] NTIA. 2021. Survey of Existing SBOM Formats and Standards. https://www.ntia.gov/files/ntia/publications/sbom_formats_survey-version-2021.pdf
[106] Phil Odence. 2023. Why you should use SPDX for security. https://www.linux.com/featured/why-you-should-use-spdx-for-security/.
[107] Marc Ohm, Henrik Plate, Arnold Sykosch, and Michael Meier. 2020. Backstabber's knife collection: A review of open source software supply chain attacks. In *DIMVA'20: 17th International Conference, Lisbon, Portugal, June 24–26, 2020, Proceedings 17*. Springer, 23–43.
[108] OpenAI. 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt.
[109] OpenSSF. 2022. Securing Critical Projects Workgroup: List of Projects Identified as 'Critical'. https://tinyurl.com/sxpeasey.
[110] Sean Peisert, Bruce Schneier, Hamed Okhravi, Fabio Massacci, Terry Benzel, Carl Landwehr, Mohammad Mannan, Jelena Mirkovic, Atul Prakash, and James Bret Michael. 2021. Perspectives on the SolarWinds incident. *IEEE Security & Privacy* 19, 2 (2021), 7–13.
[111] Shari Lawrence Pfleeger and Barbara A. Kitchenham. 2001. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes* 26, 6 (2001), 16–18.
[112] Martin Pratoussy. 2022. Estab of a new workflow to manage software vulns. https://cds.cern.ch/record/2826626/files/Report-PRATOUSSY_Martin.pdf.
[113] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356* (2022).
[114] Rezilion. 2022. Dynamic SBOM: A Comprehensive Guide. https://www.rezilion.com/blog/dynamic-sbom-a-comprehensive-guide/.
[115] Dirk Riehle and Nikolay Harutyunyan. 2019. Open-source license compliance in software supply chains. In *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability: Communications of NII Shonan Meetings*. Springer, 83–95.
[116] Guillaume Rousseau, Roberto Di Cosmo, and Stefano Zacchiroli. 2020. Software provenance tracking at the scale of public source code. *Empirical Software Engineering* 25 (2020), 2930–2959.
[117] PS Rusk. 1990. The role of the bill of material in manufacturing systems. *Engineering Costs and Production Economics* 19, 1-3 (1990), 205–211.
[118] Ryan Naraine. 2022. Big Tech Vendors Object to US Gov SBOM Mandate. https://www.securityweek.com/big-tech-vendors-object-us-gov-sbom-mandate/.
[119] Adriana Sejfia and Max Schäfer. 2022. Practical Automated Detection of Malicious npm Packages. *arXiv preprint arXiv:2202.13953* (2022).
[120] Neil Sheppard. 2023. SBOMs (Software Bill of Materials): Why Do They Matter? https://www.leanix.net/en/blog/sboms-matter
[121] Donna Spencer. 2009. *Card sorting: Designing usable categories*. Rosenfeld Media.
[122] Nathan Wintersgill Oscar Chaparro Massimilano Di Penta Daniel M German Denys Poshyvanyk Stalnaker, Trevor. 2023. Online replication package. https://github.com/TStalnaker44/boms_away_study.
[123] Xin Tan, Kai Gao, Minghui Zhou, and Li Zhang. 2022. An exploratory study of deep learning supply chain. In *Proceedings of the 44th International Conference on Software Engineering*. 86–98.
[124] Wei Tang, Zhengzi Xu, Chengwei Liu, Jiahui Wu, Shouguo Yang, Yi Li, Ping Luo, and Yang Liu. 2022. Towards Understanding Third-party Library Dependency in C/C++ Ecosystem. In *in ASE'22*. 1–12.
[125] Ann R. Thryft. [n. d.]. The Challenges of Securing the Open Source Supply Chain. https://tinyurl.com/yvsfdxd9
[126] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding sparkle to social coding: an empirical study of repository badges in the *npm* ecosystem. In *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 511–522. https://doi.org/10.1145/3180155.3180209
[127] Christopher Vendome, Gabriele Bavota, Massimiliano Di Penta, Mario Linares-Vásquez, Daniel German, and Denys Poshyvanyk. 2017. License usage and changes: a large-scale study on GitHub. *Emp. Soft. Eng.* 22 (2017), 1537–1577.
[128] Christopher Vendome, Mario Linares-Vásquez, Gabriele Bavota, Massimiliano Di Penta, Daniel German, and Denys Poshyvanyk. 2015. License usage and changes: a large-scale study of Java projects on GitHub. In *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, 218–228.
[129] 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. *arXiv preprint arXiv:2301.05362* (2023).
[130] Henry Young. [n. d.]. SBOMs: Considerable Progress, But Not Yet Ready for Codification. https://tinyurl.com/y2xzxs8m.
[131] Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams. 2023. Software Bills of Materials Are Required. Are We There Yet? *IEEE Security & Privacy* 21, 2 (2023), 82–88.
[132] Nusrat Zahan, Laurie Williams, Thomas Zimmermann, Patrice Godefroid, Brendan Murphy, and Chandra Maddila. 2021. What are Weak Links in the npm Supply Chain? *arXiv preprint arXiv:2112.10165* (2021).