A Comprehensive Study of Bills of Materials for Software Systems

Trevor Wayne Stalnaker

College of William & Mary
Williamsburg, VA

Bachelor of Science, Washington and Lee University, 2020

A Thesis presented to the Graduate Faculty
of The College of William & Mary in Candidacy for the Degree of
Master of Science

Department of Computer Science

College of William & Mary
August 2023

© Copyright by Trevor Stalnaker 2023

APPROVAL PAGE

This Thesis is submitted in partial fulfillment of
the requirements for the degree of

Master of Science

Trevor Stalnaker

Approved by the Committee, July 21, 2023

Committee Co-Chair
Denys Poshyvanyk, Chancellor Professor, Computer Science
College of William & Mary

Comittee Co-Chair
Oscar Chaparro, Assistant Professor, Computer Science
College of William & Mary

Gang Zhou, Assistant Professor, Computer Science
College of William & Mary

COMPLIANCE PAGE

Research approved by

Protection of Human Subjects Committee

Protocol number(s): PHSC-2022-07-14-15722

Date(s) of approval: 09/08/2022

ABSTRACT

Software Bills of Materials (SBOMs) have emerged as tools to facilitate the management of software dependencies, vulnerabilities, licenses, and the supply chain. Significant effort has been devoted to increasing SBOM awareness and developing SBOM formats and tools. Despite this effort, recent studies have shown that SBOMs are still an early technology not adequately adopted in practice yet, mainly due to limited SBOM tooling and lack of industry consensus on SBOM content, tool usage, and practical benefits. Expanding on previous research, this thesis reports a comprehensive study that first investigates the current challenges stakeholders encounter when creating and using SBOMs. The study surveyed 138 practitioners belonging to five groups of stakeholders (practitioners familiar with SBOMs, members of critical open source projects, AI/ML practitioners, experts of cyber-physical systems, and legal professionals), using differentiated questionnaires. We interviewed eight survey respondents to gather further insights about their experience. We identified fourteen major challenges facing the creation and use of SBOMs, including those related to the material included in SBOMs, deficiencies in SBOM tools, SBOM maintenance and verification, and domain-specific challenges. We propose and discuss six actionable solutions to the identified challenges and present the major avenues for future research and development. We hope these solutions can be adopted by the community to improve SBOM formats, tools, and adoption, and thus, enable the full potential of SBOMs.

TABLE OF CONTENTS

Acknowledgments iv

Dedication v

List of Tables vi

List of Figures viii

1 Introduction 2

2 Background & Related Work 6

3 Methodology 10

3.1 Study Design 10

3.1.1 Survey Design 11

3.1.2 Participant Identification 12

3.1.3 Survey Response Collection and Analysis 15

3.1.4 Interviews Design and Response Analysis 17

4 Results 19

4.1 **RQ**$_1$: SBOM Creation and Usage 19

4.1.1 SBOM awareness and formats 19

4.1.2 SBOM use cases, benefits, and data fields 21

4.1.3 SBOM generation process, tooling, and distribution 22

4.2 **RQ**$_2$: SBOM Challenges 23

4.2.1 C1: Complexity of SBOM specifications 23
4.2.2 C2: Determining data fields to include in SBOMs 24
4.2.3 C3: Incompatibility between SBOM standards 24
4.2.4 C4: Keeping SBOMs up to date 25
4.2.5 C5: Insufficient SBOM tooling 26
4.2.6 C6: Inaccurate and incomplete SBOM 27
4.2.7 C7: Verifying SBOM accuracy and completeness. 28
4.2.8 C8: Differences across ecosystems and communities. 29
4.2.9 C9: SBOM completeness and data privacy trade-off. 29
4.2.10 C10: SBOMs for legacy packages and repositories 30
4.2.11 C11: Inability to locate dependencies for SBOMs 30
4.2.12 C12: Unclear SBOM direction 31
4.2.13 C13: Generating global software IDs 32
4.2.14 C14: Managing SBOM versions 33
4.2.15 Challenge Relationships 34
4.3 $\mathbf{RQ}_3$: Solutions to SBOM Challenges 34
4.3.1 S1: Multi-dimensional SBOM specifications. 35
4.3.2 S2: Enhanced SBOM tooling and build system support. 36
4.3.3 S3: Strategies for SBOM verification. 38
4.3.4 S4: Increasing incentives for SBOM adoption. 39
4.3.5 S5: Improving documentation. 39
4.3.6 S6: Techniques for generating software IDs. 40

5 Threats to Validity 45

6 Conclusion 47

7 Bibliographical Notes 49

A Participant Demographics 50

B Additional Data 56

C Full Comparison with Related Works 65

D Further Details on Response Filtering 68

E Survey Questions 70

E.1 SBOM C&A 70

E.2 Critical Projects 72

E.3 Machine Learning 75

E.4 Cyber-Physical Systems 77

E.5 Legal Practitioners 78

F Image Credits 80

ACKNOWLEDGMENTS

I am immensely grateful to my co-advisors, Dr. Denys Poshyvanyk and Dr. Oscar Chaparro, without whom this thesis would not have materialized. Their guidance, expertise, and unwavering support throughout the research process has been invaluable.

In addition, I would like to express my sincere appreciation to my colleagues Nathan Wintersgill, Daniel German, and Max Penta. Their tireless effort and constructive feedback have played a pivotal role in bringing this research to fruition.

I want to thank all of my committee members for their time, guidance, and invaluable contributions to my thesis and research journey.

I would like to extend my sincere acknowledgement and gratitude to the participants of this study, including those who took the time to complete the survey and those who graciously agreed to participate in follow-up interviews. Their valuable contributions and willingness to share their insights have been indispensable to the success of this work.

Lastly, I am deeply indebted to my God, who has directed my steps and been a light on my path. He has opened so many doors and His guidance has been a constant source of strength, enabling me to make it this far in my studies.

To my parents. Thank you for always encouraging me and pushing me to do my best. I love you.

LIST OF TABLES

3.1 Survey questions for different participant groups 11
3.2 Survey Statistics 15
3.3 Abbreviated participant demographics 16

4.1 Identified challenge relationships and brief justifications 42
4.2 How each SBOM solution addresses SBOM challenges and the roles impacted by the challenges/solutions 43
4.3 Relationships between challenges, solutions, and roles. 44

A.1 Participant education by category 50
A.2 Types of software developed by participants 51
A.3 Survey participant contributions to open and closed source software . 51
A.4 Interview participant contributions to open and closed source software 51
A.5 Participant roles by survey 52
A.6 Participant countries 53
A.7 Participant backgrounds in licensing and security 53
A.8 Programming languages used by participants 54
A.9 Frequency of involvement in project releases by participants 54
A.10 Breakdown of SBOM C&A respondent roles 55
A.11 Breakdown of how participants directly contacted were identified . . . 55

B.1 Why SBOMs are used in practice 56
B.2 When should SBOMs be produced for a piece of software 57
B.3 When should SBOMs be produced for a piece of software (percentages) 57

B.4 Formats used by participants 57
B.5 Dependency tracking techniques used by participants 58
B.6 Use cases for SBOM mentioned by participants 58
B.7 Benefits of SBOM reported by participants across surveys 59
B.8 Fields to include in BOM 60
B.9 Fields to include in BOM (continued) 61
B.10 Deficiencies in SBOM standards identified by participants (by count) . 62
B.11 Challenges identified by participants (by count) 63
B.12 Solutions proposed by participants (by count) 64

C.1 SBOM study methodology and scope 66
C.2 Challenge coverage of related works 66
C.3 Challenges discussed across contemporary works 67

F.1 Attribution of symbols used 80

LIST OF FIGURES

3.1 Research methodology 11

4.1 Perceived sufficiency of SBOM tooling. 26

4.2 Graph of challenge dependencies 34

A Comprehensive Study of Bills of Materials for Software Systems

Chapter 1

Introduction

The software supply chain has increasingly grown in complexity with the proliferation of open-source software [80, 130] and AI/ML components [128, 81, 82]. Organizations and developers no longer have to rely entirely on in-house efforts to solve repetitive problems in their software. Instead, they can find and reuse open and freely-available software packages, in their desired programming language, specifically designed to accomplish the tasks. This enables them to easily create, build, and publish their software products by integrating components from a variety of vendors [73].

However, leveraging such open and freely-available packages does not come without a cost. The fate of a software product is intrinsically tied to its evolving dependencies [61]. If a dependency displays a vulnerability, then so too could the final product, potentially leading to severe consequences [113]. Furthermore, open-source packages often come with licenses that specify the terms and conditions in which they may be used, modified, and redistributed. Failing to comply with the license terms of software packages can result in severe legal and economic consequences. Ensuring that all code in a product adheres to the licenses of all its dependencies can be a complex task [133, 132, 68, 121].

In this scenario, Software Bills of Materials (SBOMs) have emerged as mechanisms that facilitate the management of software dependencies [106], leading to improved management of software vulnerabilities, enhanced license compliance, and increased transparency in the

software supply chain [105]. The advantages of SBOMs stem from their attributes, as they provide machine-readable metadata that uniquely identifies a software product and all the dependencies used in building or composing the product [106].

While SBOMs were introduced in the early 2010s [6], the 2021 US Presidential Executive Order 14028 on Improving the Nation's Cybersecurity [12] gave new momentum to SBOM formalization and adoption [11] as it required companies selling software to the US government to provide SBOMs. This was prompted by recent open source supply chain attacks, such as the SolarWinds breach [116] and critical vulnerabilities such as those affecting the Log4J library [91], which impacted a large number of users [103, 71]. SBOMs are currently championed by the US National Telecommunications and Information Administration (NTIA) [106, 104] and well-known organizations such as the Linux Foundation [8] and OWASP [4]. Significant effort has been put into promoting SBOM formats and tools that can create and process them [108], with the goal of increasing adoption and fully enabling the benefits that SBOMs offer [105].

Although organizations and developers have acknowledged the potential advantages of SBOMs and anticipate using them more frequently in the coming years [126], recent research indicates that there are concerns regarding their commitment to SBOMs and the actual benefits SBOMs bring to their projects [134]. 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 [21]. Additionally, the lack of mature tool support to fully automate the production and consumption of accurate and complete SBOMs is an important barrier to SBOM adoption [134].

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 and enable the new opportunities.

This thesis 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), including those that incorporate hardware [50], services, or AI/ML components, we distributed five distinct questionnaires to different groups of stakeholders, resulting in a total of 150 responses (84 responses indicating SBOM familiarity). The surveys targeted software practitioners familiar with SBOMs, contributors of critical open-source projects [115], AI/ML practitioners, experts in Cyber-Physical Systems (CPS) [50], and legal professionals. 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 who span the different groups. We analyzed participant responses using state-of-the-art practices for qualitative and quantitative data analysis [117, 85, 86, 87, 88, 89].

This study expands our understanding of the specific limitations and challenges of current SBOM formats and related tools. Additionally, it 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 thesis are:

- A comprehensive empirical study of SBOM adoption, challenges, opportunities, and solutions. The study targeted five groups of stakeholders according to the different types of software that SBOMs should support. The study offers greater scope and different perspectives about SBOM adoption compared to recent prior studies [134, 43, 136];

- 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.

Chapter 2

Background & Related Work

Bills of Materials (BOMs) come from the manufacturing industry and refer to the list of raw materials, components, and parts needed to manufacture an end product [100, 123]. The concept has been transferred to software systems as Software BOMs (SBOMs), which list a project's dependencies and provide provenance. There are currently three major SBOM standard formats: SPDX [23], CycloneDX [18], and SWID [78]. While the NTIA has not officially endorsed any one standard [111], SPDX was officially recognized as a standard by ISO in 2021 [77].

Component inventory, vulnerability analysis, and license compliance are some primary SBOM uses cases [66]. SPDX, supported by the Linux Foundation, began as a solution for managing open-source licenses, whereas CycloneDX, supported by OWASP, focuses primarily on security and vulnerability management.

These different design philosophies result in a few notable differences: (1) SPDX can represent and license code snippets within files, (2) SPDX supports annotation (adding comments to an SPDX document), (3) CycloneDX natively supports compositions, and (4) CycloneDX offers more robust support for vulnerability management [66, 67].

Both specifications support a variety of standard file formats. In total, SPDX supports five file types: tag/value (.spdx), JSON, YAML, RDF/xml, and spreadsheets (.xls) [9]. CycloneDX supports both JSON and XML schema as well as protocol buffers [7].

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) [54, 58, 83];
- hardware (HBOMs) [56] and firmware (FBOMs) [39, 40, 63];
- operational (*e.g.*, configuration) environments (OBOMs) [57]; &
- datasets (DataBOMs) [46] and AI models (AIBOMs) [49, 134].

This study targets specific populations of software stakeholders (*e.g.*, experts in AI and Cyber-Physical Systems) to understand needs that could be potentially fulfilled by the aforementioned BOMs.

While SBOMs have existed for some time [6, 1, 78, 48], they are only now beginning to be widely known [135, 110]. The analysis of their uses and shortcomings has been investigated only by a few recent studies [99, 134, 43, 50, 74, 51, 129], which we discuss next.

A survey from the Linux Foundation examined the current state of SBOM usage and readiness in industry [74], aiming to identify the main use cases, benefits, and unmet needs for SBOM. 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 [51]. They found that, generally, source code used in development was the least-important feature for this group, yet SBOM information was valued differently by people in 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 recent study of C/C++ library ecosystems [129], Tang *et al.* found little evidence of SBOMs being used in open-source projects. Of over 24K GitHub repositories examined, fewer than ten contained recognizable SBOMs. However, they identified that some groups may be using package manager-specific files with similar information to SBOMs as approximations of SBOM documents. These can be considered "quasi-SBOMs."

A gray literature review conducted by Zahan *et al.* examined common challenges faced by developers when using SBOMs [136]. While this work complements our own, we consider a wider array of developer categories and BOM types. A detailed comparison of the two works is found in Appendix C.

The study by Xia *et al.* [134] is the closest to our work. Xia *et al.* interviewed 17 software practitioners to derive 25 statements about SBOM practices, tools, and concerns. Then, 65 software practitioners were surveyed, indicating their agreement with the statements via Likert scales and providing explanations based on their experience with SBOMs via open-ended answers. Their results were summarized in a set of ten findings, highlighting, among others, the need for integrating SBOM formats with information exploitable in typical usage scenarios (*e.g.*, including better information about vulnerabilities), the still limited level of awareness about SBOM usage, the immaturity of existing tools, and the lack of suitable trust mechanisms. Our study extends this prior work as it: (1) includes five distinct surveys that target a wider population of software stakeholders (*e.g.*, AI/ML, CPS, and legal practitioners), (2) investigates usage, opportunities, challenges, and solutions for different types of BOMs applied to software systems, (3) analyzes the specific challenges that stakeholders face when creating and using SBOMs, and (4) discusses actionable solutions to overcome these challenges.

SBOM tooling is currently underexplored in the literature. Lin explores the use of SBOM tools for DevSecOps and software composition analysis [98]. 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 all the transitive dependencies, compared to those given by the Maven build system [43]. The tools capture a different set of dependencies for a

project, missing a large portion of the Maven dependency tree. Based on these results, Balliu *et al.* discuss open challenges for accurate SBOM generation and effective SBOM consumption. Beyond these two works, improvements to the state of SBOM tooling have largely been an open-source and community-led effort. Our study provides a more comprehensive view of different stakeholders' problems regarding SBOMs, beyond Java systems, SBOM tools, and security-related applications.

There have been a number of proposals for tracking dataset and AI model information [47, 76, 101, 70]. None of these works look specifically at applying the concept of BOMs to the data and model supply chains. The term DataBOM was introduced and discussed by Barclay *et al.* [46]. However, their work does not survey developers to determine the feasibility of the idea or which fields should be included, as we do. Potential use cases within the domain of AI/ML are mentioned, but DataBOM is 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 [49], 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 [45].

Chapter 3

Methodology

3.1 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 different groups of stakeholders, namely software developers, project leaders and contributors, AI/ML experts, CPS experts, and legal professionals.

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. 3.1).

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

Table 3.1: 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 Projects	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

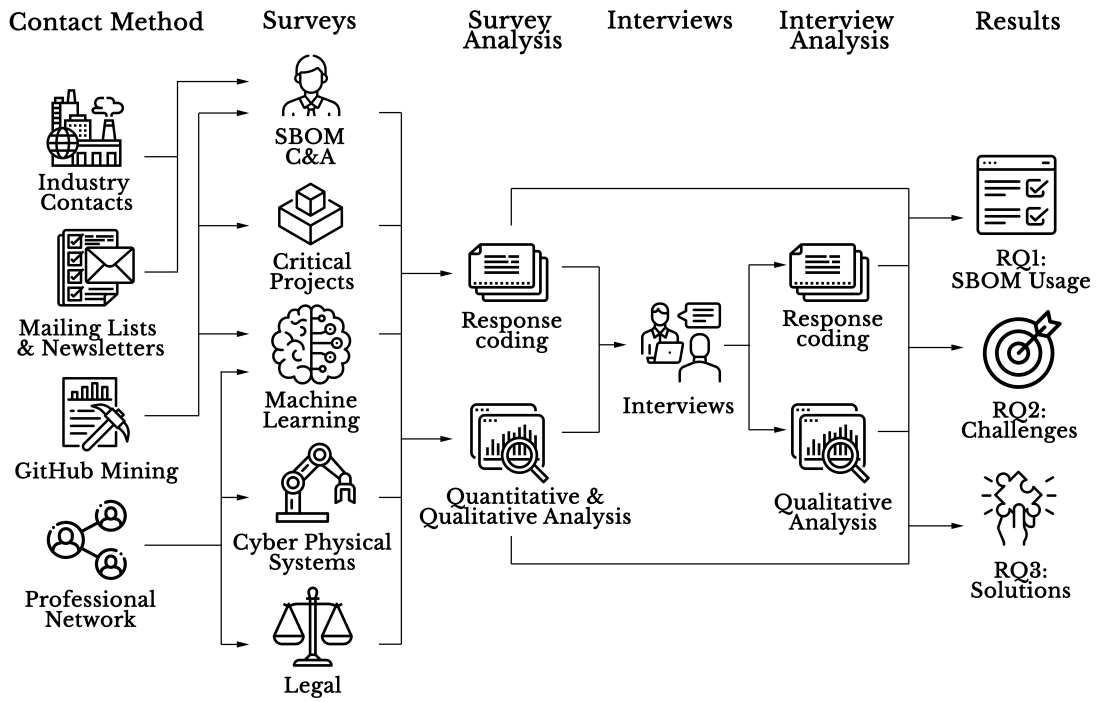

Figure 3.1: Research methodology

3.1.1 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 [72], as well as SE specific guidelines [117, 85, 86, 87, 88, 89].

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

Table 3.1 provides an overview of the information we asked for in the surveys. A detailed description of all questions can be found in our replication package [27]. The surveys contain a mix of (five-point) Likert-scale, multiple-option, and open-ended questions that asked about (see Table 3.1): 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 that fully completed the survey were entered into a lottery to win one of ten Amazon gift cards.

3.1.2 Participant Identification

We identified five groups of study participants, namely SBOM Community and Adopters (SBOM C&A), developers of critical open source (OSS) projects [115], developers and researchers of Cyber-Physical Systems (CPS) and AI/ML systems, and legal experts.

SBOM Community and Adopters (SBOM C&A). These are people who work with SBOMs in different manners [41, 50]. Contacting people who directly use SBOMs and related technologies allowed us to obtain firsthand reports of 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 mining of GitHub repositories. Combining manual effort and automated tools (based on GitHub APIs [16]), 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 about 4,423 developers' email addresses. A similar approach to identifying study participants was used by Xia *et al.* [134].

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 [25] to locate dependent projects with SBOM-related tags. These repositories and their contributors logically represent groups currently using SBOMs.

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 [37] and the OpenChain mailing lists [3].

Developers of Critical Open Source Projects. The Open Source Software Foundation's workgroup on Securing Critical Projects compiled a list of the 102 most critical software projects, comprising 564 total repositories [115]. The projects include the Linux Kernel, programming languages, popular libraries, package managers, databases, *etc.* Given their importance, we wanted to learn if and how these projects made use of SBOMs. Examining mainstream projects would also allow us to assess how SBOMs have spread beyond early-adopter communities. Given the role of SBOMs in the software supply chain, we sought to administer a targeted survey examining projects which are widely depended on, as such projects may have a greater need to 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.

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 experts that work on the development of CPS and have expertise in the interaction between hardware and software. We have identified this group of users through our professional network. Surveying this group allowed us to examine unique challenges facing the usage of both SBOMs and HBOMs, as well as how the two may interact.

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 the replication package [27]); and (ii) AI/ML experts in our academic/professional network.

Table 3.2: Survey Statistics

Survey	Full Resps	Valid Resps	Fam. w/ SBOMs	Inter-views	Role	#
SBOM C&A	179	101	61	4	P	34
Critical	22	22	13	1	C	31
ML	21	20	8	1	TM	24
CPS	6	6	1	1	E	14
Legal	1	1	1	1	SM	16
Totals	229	150	84	8	O	7

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

Machine learning components have their own supply chains, but are also increasingly included in software products. Tracking model and data provenance is essential to security (*e.g.*, model poisoning), licensing / use issues, and research. While similar in some regards, the needs, challenges, and use cases facing AI/ML developers are different from those of typical SBOM users. By surveying this group, we aimed to explore these topics.

Legal Experts. 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. There is a small pool of legal practitioners with development experience and familiarity with SBOMs, making this the hardest group to survey at scale.

3.1.3 Survey Response Collection and Analysis

Responses from the survey participants were collected using Qualtrics [34]. 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.

We directly invited more than 4.4k individuals via email to participate in the surveys and received 229 complete responses in total (see Table 3.2). These were analyzed following the procedure described below, resulting in 150 valid responses. Table 3.3 overviews the demographics for all the study participants.

Table 3.3: 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

Collected data was exported as a raw CSV and post-processed into JSON format for better readability and searchability, removing personally identifiable information for privacy. Answers for each question were extracted so that they could be considered in isolation. Associated question IDs for each response were also included for reference if needed.

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 Chapter 4.

For the open-ended questions, a coding approach was applied in line with [127]. 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.

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 [114]. 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. See Appendix D for more information.

3.1.4 Interviews Design and Response Analysis

We conducted one-hour semi-structured interviews with eight participants of the surveys (see Table 3.2), to gather deeper knowledge about their experience and responses. In agreeing to participate, respondents indicated willingness to be contacted for a follow-up interview. We selected respondents from the five surveys whose responses warranted further investigation in the form of follow-up questions and inquiries into their unique

areas of expertise. In particular, we sought interviews with respondents who indicated experience in their field regardless of their familiarity with SBOMs and gave detailed responses that highlighted interesting use cases, challenges, or potential solutions. In this way, we hoped to capture an array of perspectives both from respondents who were very familiar with SBOMs, but also from respondents who were not as familiar and had interesting perspectives on how SBOMs might affect them and their work. This resulted in 8 interviews as shown in Table 3.2.

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. The replication package contains the protocol we followed for the interviews [27].

Interviews were conducted over Zoom and recorded with the participants' permission. The recordings were transcribed using the Whisper speech recognition tool [119]. 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.

Interview participants were compensated for their time with an additional Amazon gift card.

Chapter 4

Results

We discuss the results of our study regarding SBOM creation and usage, challenges, and solutions.

56% (84/150) of the study participants are familiar with SBOMs (Table 3.2). The 22 respondents of the critical survey belong to 16 of the 102 (15.7%) critical OSS projects.

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 (32%) reported using SPDX, 8 (16%) CycloneDX, and 12 (24%) both. SWID [106] was used by only 10% (5) of 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 open source projects survey participants, 9 (41%) were unfamiliar with SBOMs and 7 (32%) were aware of SBOMs but had not yet adopted. Of those who reported using/producing SBOMs, most used package configuration files (*e.g.*, Maven or Node package files—*a.k.a.* "quasi-SBOMs" [129]). One interviewee mentioned how the limited interest in true SBOM is due to the limited tool support and the need for manually maintaining SBOMs. This is consistent with the findings of Zahan *et al.* [136].

We found limited evidence of SWID usage, and this could be in part, as one interviewee mentioned, because the SWID specifications are hidden behind an ISO paywall [78, 102]. That said, a guide to creating SWID tags is available through the NTIA [60], and we found a few usages of a concise version of SWID, called coSWID [75], in our survey.

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 [79, 118] and popular standards have been mentioned by Eggers *et al.* for the nuclear industry [62].

Of the six CPS respondents, 3 (50%) were familiar with HBOMs and 2 (33%) 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 thesis, we have also learned that CycloneDX has added a Machine Learning Bill of Materials (ML-BOM) to its specification [55].

Participants expressed that the pressure to have SBOMs is largely felt by industry and projects at the end of a supply chain. At present, there is little incentive for open-source projects towards the beginning of a supply chain to produce SBOM. Some of these 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 [105] and prior studies [134, 136], we found that security, dependency tracking [122], and licensing are the main use cases for SBOMs.

Out of 61 SBOM practitioners, 55 (91.8%) mentioned as main use case dependency management, 22 (36%) licensing concerns, and 22 (36%) software security (*e.g.*, vulnerability) management. Other responses include software versioning (14, 23%), provenance (10, 16.4%), documentation (6, 9.8%), and transparency (4, 6.5%).

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.* [136], has been addressed by NTIA [107] and our interviewees rejected the notion of security by obscurity as a solution.

When 41 SBOM producers, tool makers, and standard makers were asked which fields should be included in SBOMs, responses varied. The most common answers were general information about the software components: version number (24, 55.81%), license (22, 51.16%), component name (18, 41.86%), and a URL to the component (18, 41.86%). Notably, 13 (30.23%) respondents indicated that the SBOM should contain unique identifiers for the software component the SBOM is documenting and/or its dependencies [75, 15, 22, 5, 2, 14].

Although we found little evidence to suggest AI and DataBOMs 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, AIBOM can provide transparency into how a model was trained, providing information about its architecture, hyper-parameters, and any pre-trained base models used. A linked DataBOM would also make developers aware if a model was trained using a poisoned, biased, or illegally sourced dataset by providing provenance and usage information for all data points.

From discussions with CPS domain experts, we identified that BOMs could serve as regulatory documents for critical embedded systems (consistent with the findings of [50]), and that they could increase the transparency and reproducibility of research results in academic circles. For these tasks, the BOMs must communicate information related to the physical hardware components (part numbers, manufacturer, etc), firmware, and 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 [26]), and open source solutions (*e.g.*, ScanCode [35]).

Despite the NTIA recommendations [109], 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 out of 12 (41.67%) critical project developers mentioned SBOM distribution as a challenge moving forward. As one interviewee put it, "wherever you're getting the package, get the SBOM too." This was a view consistent across all groups surveyed.

Concerning support for DataBOMs and AIBOMs, two survey participants mentioned that Hugging Face dataset cards [65] could serve as DataBOMs, while it does not directly provide tool support. Three respondents mentioned the same service's model cards, providing similar information to AIBOMs. Other tools mentioned include DVC [29] and ML-Flow [33]. These formats fall under what we might call quasi-AIBOMs, since, to our knowledge, no formal AIBOM standards have been implemented and accepted in practice.

When 34 producers were asked in which moments of the development process SBOMs should be generated, they said: during each build (28, 80.35%), when publishing a major release (21, 61.76%), during deployment (19, 55.88%), and at the developer's discretion (7, 20.6%).

4.2 RQ_2: SBOM Challenges

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

4.2.1 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."

When support for a new use case is added to the specification, the latter becomes longer and more complicated, potentially leading to lower adoption and lower quality SBOMs. One 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]." A critical OSS infrastructure developer remarked: "Standards are too stringent, I shouldn't need to consult a lawyer to list my code inputs."

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 see all the security garbage in there. [...] Who cares what a vulnerability ID or a CPE is. [...] I don't want to have to learn all this just to be able to use the spec." However, "even if you, as a producer of an SPDX document, don't have that use case in mind, your consumers may have that use case in mind."

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".

4.2.2 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 the BOMs to link to a 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, 85%) and validation/testing data (14, 70%), preprocessing steps taken on the data (13, 65%), dataset version (13, 65%), and used optimizers/loss functions (13, 65%). When asked about fields needed in DataBOMs, they highlighted data sources (18, 90%), data transformations (18, 90%), preprocessing steps (17, 85%), dataset size (16, 80%), known/potential biases (14, 70%), and data collection procedures (14, 70%).

Of the 6 CPS experts we surveyed, 3 (50%) expressed a need for hardware part numbers, 2 (33.33%) testing and quality assurance data, 1 (16.67%) system deployment information, 1 (16.67%) manufacturer information and location, and 1 (16.67%) known limitations about components (*e.g.*, if they are not suitable for certain tasks due to security risks or hardware constraints).

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

4.2.3 C3: Incompatibility between SBOM standards

Responses show that competing standards confuse developers. When consuming SBOMs, 23.33% of the SBOM practitioners stated that different standards pose a challenge, due to interoperability issues between standards and inconsistency between standards and tooling.

Despite this, one expert 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." Later adding that, "supporting too many use cases, too many possibilities, is not a good thing. So, simplification, slashing through, having fewer possibilities and having only one way to express something would be something that would help a lot."

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.

4.2.4 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 [17]. These changes can happen suddenly. For example, if a developer adds an application to a container, the number of dependencies can dramatically increase [84]. Also, changes may even occur asynchronously from those to the software. For example, new vulnerabilities may be discovered in existing dependencies, which could affect SBOMs (if they contain vulnerability information)— 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 three critical project developers that consume SBOM, one mentioned

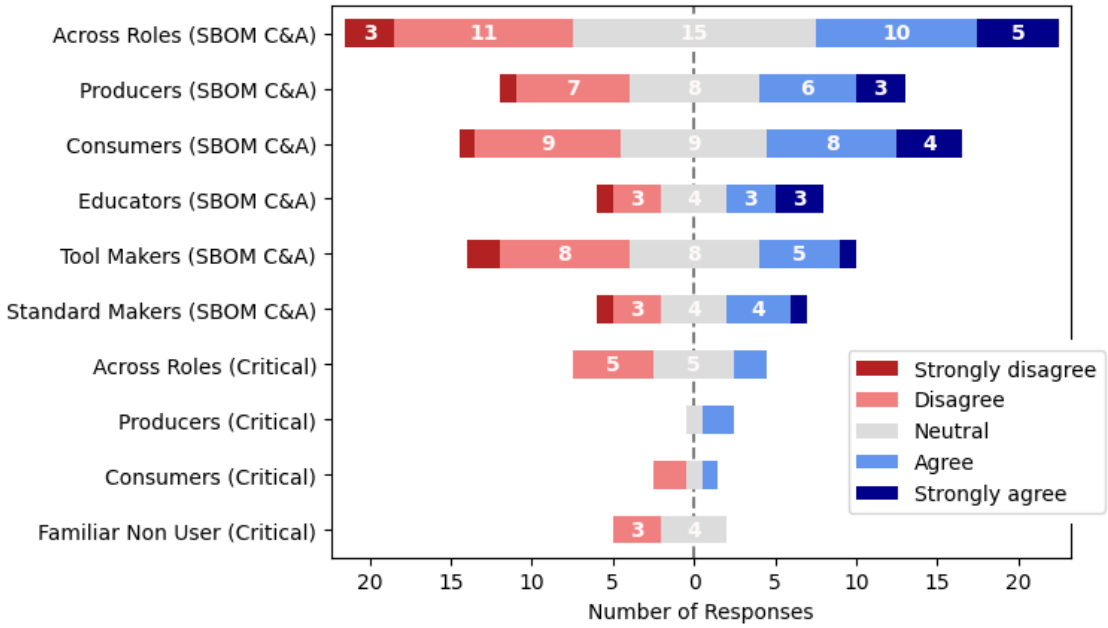

Figure 4.1: Perceived sufficiency of SBOM tooling.

difficulties in keeping SBOMs up-to-date. This motivates a need for tools which can dynamically update SBOMs as changes occur [120].

4.2.5 C5: Insufficient SBOM tooling

Figure 4.1 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. This suggests that either such tooling does not yet exist, is insufficient, or is obscure. All three present challenges to SBOM usage. 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 couldn't "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 [...]."

While respondents identified various tools to produce SBOMs, in our interviews, multiple respondents expressed a comparative lack of tool support for SBOM consumption.

4.2.6 C6: Inaccurate and incomplete SBOM

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 practitioner survey mentioned poor quality SBOMs as one of the challenges they had faced in using SBOMs. 25% of the consumers from the OSS critical projects 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 indirect (transitive) dependency information, including the URLs of their sources. The legal expert we interviewed mentioned that, in his/her experience, this condition is rarely met.

One interviewee suggested that if stakeholders perceive BOMs to be just another requirement to fulfill, developers will likely cut corners by filling in incorrect defaults or incorrectly copy-pasting from older BOM documents: "What are they going to do with the version number? They're going to type 'one'. Oh... the system didn't let me enter it, so [I'm] going to type 'two'."

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."

Participants also discussed "false positives" in BOMs. For example, just because your project uses a dependency and that dependency has a vulnerability, it does not follow that your project will necessarily be impacted. It depends on the usage context of that dependency. Determining if a project is actually impacted is a more difficult problem and will require more sophisticated tooling.

4.2.7 C7: Verifying SBOM accuracy and completeness.

33% of the OSS critical infrastructure contributors mentioned how SBOM verifiability is a major challenge. This was also reported by 3 participants of the SBOM practitioner survey.

During an interview, one participant said, "I think [the consumer] inherently has to trust the content of the SBOM as accurate [...] If the consumption tools can't really trust the content, then they're going to have to go get the data themselves from somewhere else, which is what the generation tools have to do. So then it kind of defeats the purpose of the file format in the first place."

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.

4.2.8 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 critical project 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 project developers surveyed who were familiar with SBOMs expressed a need for more language specific tooling.

4.2.9 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 person-

ally identifiable, private, sensitive, or proprietary information. This challenge also depends on how the data was collected and then processed. CPS respondents also mentioned privacy concerns in BOMs, as CPS may actively collect and process private and sensitive data from the environment. Corporate entities have also expressed concern over exposing proprietary information through SBOM [21].

4.2.10 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 [69] or the banking industry [97]. A 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"

4.2.11 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 expert 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 [93, 92, 94, 90, 42, 52, 64, 137, 96, 125] and are commonly removed from package managers once detected [53]. Since CVEs are for vulnerabilities [13], 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.

Packages could also be removed by their creators if they are no longer being maintained or if there is a move to a new commercial, closed-source version of the software. Packages may also be removed for violating laws or the terms of use for the service hosting them. For example, after an RIAA DMCA notice, GitHub removed the popular YouTube-dl repository [95].

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.

4.2.12 C12: Unclear SBOM direction

US Presidential Executive Order 14028 on Improving the Nation's Cybersecurity requires companies selling software to the US government to provide corresponding SBOMs. This has created an important incentive at the end of the software development chain to create and maintain SBOMs; however, the results shown demonstrate that there are still many stakeholders that are yet to adopt (and in some cases know about) SBOMs. It is desirable that developers of open source libraries create and maintain SBOMs; however, while those consuming these libraries will benefit from such SBOMs, it is less clear how the developers of the libraries will benefit. Creating SBOMs is not trivial and requires effort. Without proper incentive, most developers forgo this effort.

The Information Technology Industry Council [21] wrote an open letter in response to recent pushes from the US federal government to mandate SBOMs [124]. 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, and the protection of confidential or proprietary information, all issues mentioned by respondents during our study. Though, many of these concerns have also been addressed by the NTIA [107].

This suggests a fear that the work required to create and maintain SBOMs would outweigh their benefits. As one of our experts 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. This excerpt from an ML survey response succinctly describes the challenge, "A major risk is that the BOM becomes 'yet another' bureaucratic nightmare involving more and more documents to fill out, without actually solving the underlying problem"

Lastly, as we were reminded by numerous respondents, SBOMs are still a young 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?"

4.2.13 C13: Generating global software IDs

To make SBOMs actionable, their dependencies must have globally unique identifiers and name collisions must be properly handled. One interviewee mentioned: "If you're on the receiving end and trying to make sense, or trying to aggregate and combine the data from this different source, you're going to need some kind of identifier."

When asked about generating globally unique IDs, One standard maker described generating globally unique software IDs as "our hardest problem to solve."

This is further motivated in that 30% of SBOM practitioners listed software identifiers as a required SBOM field. For comparison, that is only slightly less than the 32% that listed dependencies as a must. For both AI and DataBOMs, 50% of ML practitioners, mostly unfamiliar with SBOM, selected unique identifiers as must-have fields.

Due to the dynamic nature of security vulnerabilities, respondents also mentioned that SBOM should link to external vulnerability databases (*e.g.*, the CVE) rather than describing vulnerabilities. Globally unique IDs are essential for this mapping to be effective.

Respondents also mentioned the importance of having an accessible algorithm to transparently generate a software ID instead of using arbitrary IDs. While solutions in different standards exist, they are not yet interoperable.

4.2.14 C14: Managing SBOM versions

SBOMs should have unique version identifiers just like the software that they represent. These SBOM versions are not inherently tied to changes in the underlying software however, but instead to changes made to the SBOM document itself. Given that current tooling often misses or inaccurately fills certain fields, it is sometimes necessary to manually and retroactively correct an SBOM. At this point, there are two SBOMs, nearly identical, that represent the same build. It becomes crucial to have mechanisms to properly and appropriately maintain and manage these SBOM versions. One interviewee described the problem: "In theory, you generate a new globally unique ID every time you produce an SBOM. What I find in practice is most people don't do that. [...] According to the spec, it's supposed to be absolutely unique. You change one character in that SBOM, you should change the globally unique ID for that." Without these globally unique identifiers it can become challenging for clients to determine which version of an SBOM they have on file.

4.2.15 Challenge Relationships

In this section, we explore the relationships between identified challenges.

The challenges enumerated previously do not exist in a vacuum, but instead feed into each other, creating an interconnected web of dependencies as seen in Figure 4.2. That is, that some challenges cause or exacerbate others. We observed many of these connections early in our investigation, but only formalized them after distilling our list of main challenges. We did this through rigorous discussion and analysis, focusing primarily on the root causes of challenges, but also considering the potential impact challenge resolutions might have on the remaining challenges.

The discovered challenge relationships are described in Table 4.1 and visualized in Figure 4.2. Ultimately, knowing the dependency relationships between challenges can help researchers and practitioners better decide where to focus their future efforts.

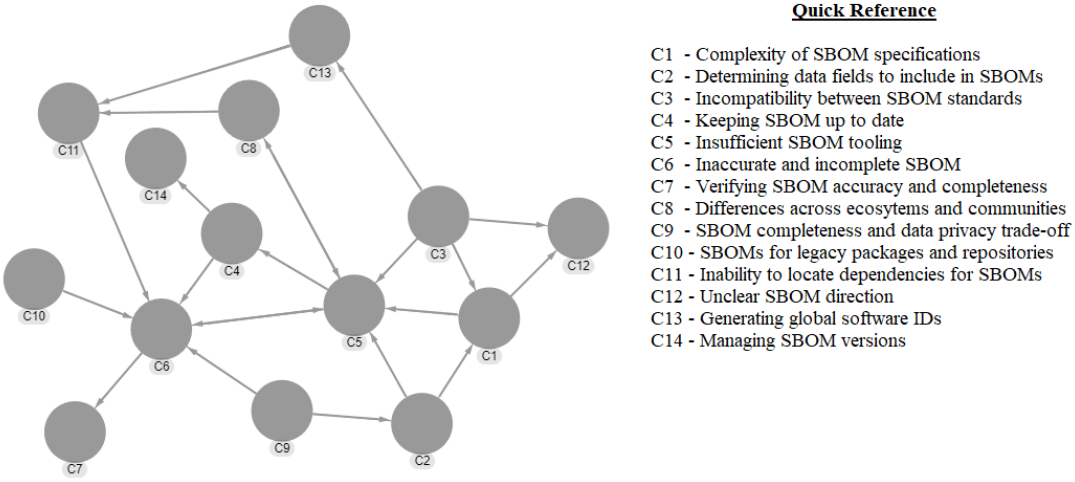

Figure 4.2: Graph of challenge dependencies

4.3 RQ_3: Solutions to SBOM Challenges

In this section, we discuss solutions for the identified challenges.

Section 4.3 shows how the solutions proposed in this section address some of the challenges identified in Item $\mathbf{RQ}_2$. The outlined solutions do not address (C3), (C10), or (C11): these will require additional research to mitigate effectively, but insights and potential directions are discussed in their challenge descriptions. **??** provides a comprehensive mapping from solutions to challenges..

4.3.1 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 being 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 [10], but including fields tailored for each of these results in cluttered specifications (see 4.2.1). In interviews, we learned that the SPDX team is working on *profiles* [112] which define the fields required in an SBOM document meant for a specific use case. This will allow producers to create SBOMs tailored for their use case without worrying about fields that to them are irrelevant. 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 [59]. One respondent highlighted additional limitations imposed by certain types of software: "You can't use SPDX with firmware as it's just too large."

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.

4.3.2 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 source tools [can address current deficiencies in tooling]." These libraries would serve as the foundation 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 tools available. 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 projects 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.

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."

Developing well-maintained and easy-to-use SBOM libraries that can be leveraged by the community can boost community engagement in delivering improved SBOM tools.

Furthermore, existing build systems (*e.g.*, Maven or Gradle) should be made SBOM-aware: capable of reading and generating SBOMs. As one practitioner put it, "one 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." Including SBOM generation functionality in build tools would more easily facilitate the update of SBOMs (C4) and their versioning (C14).

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 [20]. Through integration with GitHub's Dependency Graph tool [38], 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 the information required during generation. "I imagine [...] that eventually there'll be [...] something built into TensorFlow or PyTorch [...] that outputs a document in a JSON file [...] that tells you the key elements [like] the hyper-parameters."

4.3.3 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) [36, 28], 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 [32]. According to our legal expert, issues of liability would likely only arise if proprietary software or a 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, "central authorities have never seemed to work too well in our industry, you know, just getting everybody to adopt it", and so a more practical solution may be needed. 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.

4.3.4 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 [131]) (C12).

Similar to Executive Order 14028, other stakeholders could require their participants to provide SBOMs. For example, 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, not only for those that create the software, but also for those that consume it. 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.

4.3.5 S5: Improving documentation.

Our results show that, while documentation and educational resources exist for SBOM, they are not sufficient to address the current complexity of the specifications. One interviewee explained, "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."

The documentation for specification should be designed with end users in mind. The number of potential use cases and their associated fields make the specifications daunting (C1). One interviewee explained that, "[...] if all you care about is security, you know, you can look that up very, very quickly and not be bothered by having to learn all these extraneous things that don't have anything to do with your use case."

Better marketing and more effective communication about SBOMs is required. Effort needs to be invested in distinguishing SBOMs from other dependency-tracking systems and quasi-SBOMs, or making it clear how these can technologies can interoperate. With recent advances in large language models, it is also possible to create a system that provides clear, inter-active, and more user-friendly specifications [24, 19]. Such a system could be used to retrieve facts about a specification on demand and provide SBOM examples for different usage scenarios and contexts.

Improved documentation and educational resources are likely to improve SBOM adoption (C12).

4.3.6 S6: Techniques for generating software IDs.

There are two possible solutions to this problem (C13). First, registration systems have the benefit of allowing the inclusion of additional information, but as one expert said, "I think the problem with the more registration oriented IDs is it's just not getting adopted. Or it's getting adopted in the ways that are [like] the CPEs, you know, you'll find five CPEs for the exact same software, and that can be a little confusing."

Then, there are the machine-generated IDs. Based on the NTIA's document on SBOM versions [111] and our expert interviews, we have identified the five most common software identity formats. These are: Concise SWID Tags (CoSWID) [75], Common Platform Enumeration (CPE) [15], Package-URLs (purl) [22], SoftWare Heritage persistent IDentifiers (SWHID) [5], and GitOIDs [2, 14].

One interviewee discusses purl as a solution, "But it [package url] helps a lot to have a seamless way to identify a package just by observing the code, in most cases. And once

you have that, it becomes easy to relate that to a security database, a dependency tree, or anything else."

As is the case with SBOM, it is unlikely that a single software identification standard will emerge or win out over the others. Instead, we must live in a world where all the standards can successfully and harmoniously coexist.

Table 4.1: Identified challenge relationships and brief justifications

Contributing Challenge	Dependent Challenges	Brief justification
(C1) Complexity of SBOM specifications	(C5) Insufficient SBOM tooling	It is difficult to create tools because of the complexity of the specifications
	(C12) Unclear SBOM direction	Complex, hard to understand specs lead to lower adoption and more stakeholder confusion
(C2) Determining data fields to include in SBOMs	(C1) Complexity of SBOM specifications	Increasing the number of fields may improve SBOM quality, but will increase the complexity of the spec
	(C5) Insufficient SBOM tooling	Tools need to know which fields they should support
(C3) Incompatibility between SBOM standards	(C1) Complexity of SBOM specifications	Incompatibilities between tools and standards exacerbate the complexity of individual specifications
	(C5) Insufficient SBOM tooling	Incompatibilities between standards pose a challenge to tool creation
	(C12) Unclear SBOM direction	Incompatibilities between tools and standards contributes to lower adoption and more stakeholder confusion
	(C13) Generating global software IDs	Different standards have different ways of generating or representing global IDs
(C4) Keeping SBOMs up to date	(C6) Inaccurate and incomplete SBOM	If an SBOM is not up to date, then the information that it contains in inaccurate
	(C14) Managing SBOM versions	As the software is updated or changes are made to correct the SBOM, those SBOM versions also require management
(C5) Insufficient SBOM tooling	(C4) Keeping SBOMs up to date	Because sufficient tooling for dynamic or runtime BOMs does not exist, it is difficult to keep BOM up to date
	(C6) Inaccurate and incomplete SBOM	Poor tooling produces BOMs of poor quality
	(C8) Differences across ecosystems and communities	Differing levels of tooling support can exacerbate differences across ecosystems
(C6) Inaccurate and incomplete SBOM	(C5) Insufficient SBOM tooling	Consumption tools suffer when the BOMs they consume are inaccurate or incomplete
	(C7) Verifying SBOM accuracy and completeness	If BOMs can be inaccurate or incomplete, it becomes necessary to validate the correctness of BOMs
(C7) Verifying SBOM accuracy and completeness	-	-
(C8) Differences across ecosystems and communities	(C5) Insufficient SBOM tooling	Some ecosystems provide better resources for tools such as manifest files or package managers
	(C11) Inability to locate dependencies for SBOMs	It is harder to locate dependencies in some language ecosystems
(C9) SBOM completeness and data privacy trade-off	(C2) Determining data fields to include in SBOMs	When deciding fields to include, privacy concerns must be considered
	(C6) Inaccurate and incomplete SBOM	BOMs may be incomplete as a result of protecting private or proprietary information
(C10) SBOMs for legacy packages and repositories	(C6) Inaccurate and incomplete SBOM	Projects that rely on legacy software without BOM are likely to have inaccurate or incomplete BOM
(C11) Inability to locate dependencies for SBOMs	(C6) Inaccurate and incomplete SBOM	The inability to locate a project dependency results in an incomplete BOM
(C12) Unclear SBOM direction	-	-
(C13) Generating global software IDs	(C11) Inability to locate dependencies for SBOMs	One reason developers may be unable to find a software dependency is an ambiguous identifier
(C14) Managing SBOM versions	-	-

Table 4.2: How each SBOM solution addresses SBOM challenges and the roles impacted by the challenges/solutions

(S1) Multi-dimensional SBOM specifications		
- Structure SBOM specifications considering three dimensions: (1) use cases, (2) types of software, and (3) amount of information needed (i.e., information levels) - 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	SBOM specifications for a particular type of software or use case would be shorter and not cluttered with seemingly superfluous information. Users would easily identify and access relevant specification terms for their use cases, software, and information needs.	P, C, TM, E, SM
(C2) Determining data fields to include in SBOMs	New SBOM fields could be added for special use cases and software types without cluttering the specification for other cases/systems. Information levels would determine which SBOM data fields would be optional, recommended, and mandatory.	P, C, TM, SM
(C6) Inaccurate and incomplete SBOMs	SBOMs will be less likely to be incomplete if component information that is hard to find or irrelevant for a use case or software type is not required to be included in the SBOM.	P, C, TM
(C9) SBOM completeness and data privacy trade-off	SBOM specifications 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 tooling should be compatible with build and continuous integration systems and able to automatically create/update SBOMs at each software build. This would keep SBOMs up to date with the project.	P, C, TM
(C5) Insufficient SBOM tooling	More and improved SBOM tooling (libraries, frameworks, and tools) would be developed: supporting additional programming languages and software binaries, and seamlessly integrated with build systems, continuous integration systems, and AI/ML frameworks.	P, C, TM
(C6) Inaccurate and incomplete SBOMs	SBOM tooling integrated with software builds and AI/ML frameworks would create SBOMs with the dependencies that are actually used in binaries, releases, and models.	P, C, TM
(C7) Verifying SBOM accuracy and completeness	Base libraries/infrastructure could help verify that SBOMs created from source code and binaries contain the same dependency information.	P, C, TM
(C8) Differences across ecosystems and communities	More and improved SBOM tooling (for more prog. languages and supporting build/CI systems and AI/ML frameworks) would lead to improved SBOM usage/production across languages and ecosystems.	P, C, TM
(C14) Managing SBOM versions	SBOM tooling incorporated into build systems should automatically generate unique identifiers for each SBOM version	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 the content of SBOMs would focus on 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 (SBOM mandates, increased SBOM adoption, easier SBOM consumption, code repository promotion/integration) would drive further development to create and improve SBOM tooling.	P, C, TM, E, SM
(C12) Unclear SBOM direction	SBOM mandates and improved SBOM promotion and education materials would clarify SBOM benefits and usage costs to practitioners. Increased SBOM incentives, potentially leading to improved tooling, would enable further involvement and awareness of the open-source community in SBOM creation/usage/promotion. This could help clarify and strengthen the direction of SBOM usage.	P, C, TM, E, SM

(S5) Improving documentation		
- Provide better, more accessible educational materials for SBOM - Design documentation with the end user in mind, not just focusing on standard makers - Better marketing to showcase the potential benefits of SBOM and its differences compared with quasi-SBOM - Apply state-of-the-art LLM technology to create interactive and responsive documentation		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C1) Complexity of SBOM specifications	The underlying complexity of the specifications may remain unchanged, but improved documentation and available resources will make them easier to learn, use, and understand.	P, C, TM, E, SM
(C12) Unclear SBOM direction	Improved documentation will lower the barrier to entry for SBOM and encourage adoption. Clearer and more user-friendly documentation will help clarify current SBOM directives.	P, C, TM, E, SM

(S6) Techniques for generating software IDs		
- Registering software with a third party system. - Using currently available standards and techniques for generating unique identifiers. - Creating and maintaining interoperability between identifier types		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C13) Generating global software IDs	Consistently applying available technology and techniques may be enough to solve the problem of generating globally unique software IDs.	P, C, TM

Table 4.3: Relationships between challenges, solutions, and roles.

Challenge	**Solved by**	**Depends on**	**Affected**	**Who can fix it**
C1	S1, S5	C2, C3	P, C, TM	SM, E
C2	S1	C9	P, C, TM	SM
C3	-	-	P, C, TM	SM
C4	S2	C5	P, C	P, TM
C5	S2, S4	C1, C2, C3, C6, C8	P, C	TM
C6	S1, S2, S3	C4, C5, C9, C10, C11	P, C, TM	P, TM
C7	S2, S3	C6	P, C	TM
C8	S2	C5	P, C	TM
C9	S1	-	P, C	SM, TM
C10	-	-	P	-
C11	-	C8, C13	P	-
C12	S4, S5	C1, C3	P, C, TM	TM, SM, E
C13	S6	C3	P, C	TM
C14	S2	C4	P, C	TM

Role: P=Producer, C=Consumer, TM=Tool Maker, E=Educator, SM=Standard Maker

Chapter 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 [44], 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 experts (belonging to different areas) who may or may not use SBOMs firsthand.

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 experts 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's 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 address bogus and non-truthful answers by removing responses that were clearly copied from search results, produced by generative AI, or were otherwise nonsensical, they remain a well-accepted risk in this kind of study.

Chapter 6

Conclusion

This thesis 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 from the AI/ML and cyber-physical systems, as well as developers of OSS core infrastructure systems and legal expert 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. Also, 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, the 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 [27].

Chapter 7

Bibliographical Notes

The paper supporting the content of this thesis was written in collaboration with other members of the SEMERU research lab at William & Mary and researchers from the University of Sannio and the University of Victoria. It is currently under review for publication.

Stalnaker, T., Wintersgill, N., Chaparro, O., Penta, M., German D., & Poshyvanyk, D. (2023, March). BOMs Away! Inside the Minds of Stakeholders: A Comprehensive Study of Bills of Materials for Software Systems. Under Second Round Review.

Appendix A

Participant Demographics

When analyzing these charts, it is important to remember that not all participants were shown the same questions. Questions were presented based on previous survey answers. This means that the total responses for any given question will likely not be equivalent to the total number of survey respondents. Additionally, other questions allowed for multiple selections.

Table A.1: Participant education by category

Survey	High School	Some College	Bachelors	Masters	PhD cand.	PhD	Total
SBOM C&A	1	7	27	20	0	6	61
Critical	2	5	8	3	0	4	22
ML	0	2	5	2	1	10	20
CPS	0	0	1	3	0	2	6
Legal							
Total	**3**	**14**	**41**	**28**	**1**	**22**	**109**

Table A.2: Types of software developed by participants

Software Type	**SBOM C&A**	**Critical Projects**	**Total**
Web applications	38	15	53
Desktop applications	28	10	38
Middleware	26	8	34
Mobile applications	18	6	24
Development tools	17	10	27
Databases	15	6	21
Operating systems	9	6	15
Server applications	1		1
Scripts	1		1
Platforms	1		1
SCADA	1		1
Firmware / Embedded		2	2
VPN		1	1
Package Manager		1	1

Table A.3: Survey participant contributions to open and closed source software

Survey	Open Source	Closed Source	Both	Total
Initial	7	6	37	50
Critical Projects	9	1	12	22
Machine Learning	4	2	14	20
Cyber Physical	-	-	-	-
Legal	-	-	-	-
Total	**20**	**9**	**63**	**92**

Table A.4: Interview participant contributions to open and closed source software

Survey	Open Source	Closed Source	Both	Total
Initial	0	0	4	4
Key Projects	1	0	0	1
Machine Learning	1	0	0	1
Cyber Physical	-	-	-	-
Legal	-	-	-	-
Total	**2**	**0**	**4**	**6**

Table A.5: Participant roles by survey

Role / Survey	SBOM C&A	Critical Projects	ML	CPS	Totals
Programmer	18	9	3	1	31
Project Lead	9	6	1	1	17
Consultant	7	2			9
DevOps Engineer	7	1			8
IT Manager	3	1			4
CTO	3				3
Project Manager	3	1			4
Licensing Officer	1				1
OSPO Member	1				1
Researcher	2		4	1	7
Tester	1				1
VP	1				1
Security Team	1				1
Many Roles	1				1
Sr Director of Cybersecurity		1			1
Security Analyist	1				1
Engineering Leadership	1			1	2
Government Affairs	1				1
ML/DL Engineer			4		4
Data Scientist			3		3
Data Engineer			1		1
Assistant Professor			1		1
Hobbyist			1		1
Professor			1	1	2
Director			1		1
Educator		1			1
Senior Developer				1	1
					0
Total	**61**	**22**	**20**	**6**	**109**

Table A.6: Participant countries

Country	SBOM C&A	Critical Projects	ML	CPS	Totals
America	24	2	11	2	39
United Kingdom	4				4
Global	10	14	4		28
Germany	3	2		1	6
Italy	2		1		3
Canada	2		1		3
Switzerland	2	1		1	4
India	2				2
Latvia	1				1
Singapore	1		1		2
Japan	1				1
Portugal	1			1	2
Lithuania	1				1
New Zealand	1				1
Brazil	1				1
Czech Republic	1				1
Netherlands	1				1
Spain	1			1	2
Turkey	1				1
Norway	1				1
France		1			1
Taiwan		1			1
Sweden		1			1
Nigeria			1		1
Israel			1		1
Total	**61**	**22**	**20**	**6**	**109**

Table A.7: Participant backgrounds in licensing and security

Background		SBOM C&A	Critical Projects	ML	CPS	Totals
Computer Security	Formal	40.98%	18.18%	35%	0%	**33.03%**
	Informal	54.1%	59.09%	20%	83.33%	**50.46%**
	No	4.92%	22.73%	45%	16.67%	**16.51%**
Software Licensing	Formal	22.95%	4.55%	5%	0%	**14.68%**
	Informal	62.3%	72.73%	50%	83.33%	**63.3%**
	No	14.75%	22.73%	45%	16.67%	**22.02%**

Table A.8: Programming languages used by participants

Language	SBOM C&A	Critical Projects	ML	CPS	Total
assembly	1		1		2
bash	1		1		2
C / C++	17	16	8	5	46
C#	5	3	3	1	12
dart	2				2
delphi	1				1
elixir	1				1
erlang		1			1
flutter	2				2
glsl		1			1
go	6	1			7
groovy	1				1
hcl	1				1
java	19	9	10	5	43
javascript	22	12	5	1	40
kotlin	2	1			3
objective-c		2			2
perl		2			2
php	1	1			2
python	28	13	20	3	64
r				1	1
ruby	3	5			8
rust	2	2	1		5
scala			1		1
sql	1				1
swift	1	2			3
typescript	1	2	1		4

Table A.9: Frequency of involvement in project releases by participants

Release Frequency	SBOM C&A	Critical Projects	ML	CPS	Total
Never	10%	9.09%	10%	16.67%	10.2%
Annually	10%	13.64%	30%	50%	17.35%
Quarterly	30%	50%	45%	0%	35.71%
Monthly	24%	4.55%	10%	16.67%	16.33%
More Frequently	26%	22.73%	5%	16.67%	20.41%

Table A.10: Breakdown of SBOM C&A respondent roles

Role	Count
P	8
C	8
TM	2
E	3
SM	3
O	5
C-E	1
C-E-P	1
ALL	6
ALL-O	1
C-P	6
C-P-TM	3
C-SM-P-TM	2
C-SM-TM	1
C-TM	1
E-P	1
P-O	1
P-TM	4
SM-TM	3
C-E-P-TM	1

Table A.11: Breakdown of how participants directly contacted were identified

Survey	Mined	Professional Contact	Total
SBOM C&A	2253	0	2253
Critical Projects	901	0	901
ML	1269	41	1310
CPS	0	10	10
Legal	0	2	2
Total	**4423**	**53**	**4476**

Appendix B

Additional Data

Here I provide additional, more granular data collected during surveys and interviews.

Table B.1: Why SBOMs are used in practice

Reason / Role (by Question ID)	C2	P3	T3	Critical C1	Critical P3
To monitor my project's dependencies	18	16	14	1	3
To keep track of dependency vulnerabilities	25	20	20	2	3
Because my organization asks me to	4	2	1		1
To make it easier to understand dependencies in complex projects	14	10	11	2	2
To meet regulatory requirements	10	9	9		
To meet software licensing requirements	16	10	14	1	1
To learn more about other projects	11	0	6		
To proactively identify replacements for components that reach end-of-life	11	7	6	2	1
To provide others with information about my project		13			3
It's default		1			
Procurement		1			
None			1	1	
Reasearch and testing	1	1	1		
Other (please specify)	1				
Total participants polled	**31**	**20**	**24**	**4**	**3**

Table B.2: When should SBOMs be produced for a piece of software

When / Survey	SBOM C&A (P2)	Critical Projects	**Total**
During each build	28	3	**31**
Publishing major release	21	2	**23**
During each deployment	19	2	**21**
At devs discretion	7	2	**9**
During project planning	6	0	**6**
Other	3	0	**3**
Total Devs Polled	**34**	**3**	**37**

Table B.3: When should SBOMs be produced for a piece of software (percentages)

When	**SBOM C&A (P2)**	**Critical Projects**	**Total**
During each build	82.35%	100%	83.78%
Publishing major release	61.76%	66.67%	62.16%
During each deployment	55.88%	66.67%	56.76%
At devs discretion	20.59%	66.67%	24.32%
During project planning	17.65%	0%	16.22%
Other	4.86%	0%	4.83%

Table B.4: Formats used by participants

Question	**SPDX**	**CycloneDX**	**SWID**	**Other Format**	**Quasi SBOM**	**Spreadsheet**	**Tool**	**None**	**Generic**
SBOM C&A (S2)	62.22%	44.44%	11.11%	4.44%	8.89%	6.67%	2.22%	0%	0%
SBOM C&A (T2)	79.17%	62.5%	12.5%	8.33%	4.17%	0%	4.17%	0%	0%
SBOM C&A (E4)	78.57%	57.14%	7.14%	7.14%	7.14%	0%	0%	0%	28.57%
Critical Projects (FS1)	20%	0%	0%	0%	60%	0%	0%	20%	0%
Critical Projects (FN1)	85.71%	14.29%	14.29%	0%	0%	0%	0%	14.29%	0%
ML (B2)	50%	100%	0%	0%	0%	0%	0%	0%	0%
CPS (HB3)	0%	0%	0%	50%	0%	50%	0%	0%	0%
Across Surveys	**66.67%**	**46.46%**	**10.1%**	**6.06%**	**9.09%**	**4.04%**	**2.02%**	**2.02%**	**4.04%**

Table B.5: Dependency tracking techniques used by participants

Method	SBOM C&A (SC9)	Critical Projects (S4)	Critical Projects (NU1)
Package manager	50%	45%	56.25%
Package file		30%	12.5%
No technique		25%	12.5%
README		10%	6.25%
No dependencies	1.92%	5%	
Tool assisted	23.08%	5%	31.25%
Manually tracked	3.85%	5%	
Issue trackers			6.25%
Wiki tools			6.25%
Ad hoc	1.92%		6.25%
SBOM	17.31%		
GitHub	5.77%		
Call graph	3.85%		
Database	1.92%		
Not addressed	1.92%		
Consult experts	1.92%		

Table B.6: Use cases for SBOM mentioned by participants

Purpose / Survey	SBOM C&A	CPS (O2)
Dependency tracking	91.8%	50%
Licensing	36.07%	0%
Security	36.07%	16.67%
Versioning	22.95%	50%
Provenance	16.39%	50%
Documentation	9.84%	0%
Transparency	6.56%	0%
Machine readable	6.56%	0%
Compliance	4.92%	16.67%
Relations in system	3.28%	0%
Tool tracking	3.28%	0%
Debugging	1.64%	0%
Language	1.64%	0%
Accountability	1.64%	0%
Managing compatibility	0%	16.67%
Provide intended usage information	0%	16.67%

Table B.7: Benefits of SBOM reported by participants across surveys

Benefits	C&A (C3)	C&A (P4)	Critical (C2)	Critical (P4)	ML (C1)	ML (D7)	Total
Dependency tracking	7	5	3	3			18
Vulnerability detection	3	9	1	3	5	8	29
Provides consistency	1						1
Licensing		6	2	2			10
Satisfied requirements		2		1			3
Increased adoption		1					1
Visibility / transparency	1	1			1	1	4
Efficiency		2			1		3
Better code review		1		1			2
Fast and easy	2					1	3
Verification	1						1
Improved quality		1					1
Reproduciblity		1			6	6	13
Problem location					3		3
Detect bias					3		3
Improved iteration					2		2
Trust					2	1	3
Open access					1		1
Teaching					1		1
Unsure					1	1	2
Best practices					1		1
Verify usability					1		1
Example model usage					1	1	2
Automation					1		1
Continuous integration					1		1
Better collaboration					1		1
Compliance					1	1	2
Better data selection						2	2
Data versioning						2	2
Increased model understanding						1	1
Data analysis						1	1
Better documentation						1	1
Validation and testing						1	1
Protect IP						1	1
Prevent data abuse						1	1
Better labeling						1	1
Guarantees on performance						1	1
Total participants	11	16	4	3	20	20	74

Table B.8: Fields to include in BOM

Field	SBOM C&A	Critical Projects	ML (AI2)	ML (D3)	CPS	Total
Version	24		13	15	5	57
License	22		11	10	3	46
Name	18				2	20
URL	18				2	20
Dependencies	14		15		3	32
Identifier	13		10	10	1	34
Vulnerabilities	10			13	1	24
Creation time	10		5	12	1	28
Author / Vendor	10			11	3	24
Organization	9					9
Checksum	6				1	7
Use case	3					3
Ecosystem	3					3
Description	2		11	13	1	27
Linking	2					2
Minimum elements	2					2
Tool support	2				1	3
Modified	1					1
Mitigations	1					1
Performance requirements	1		8		2	11
AI related	1					1
Support	1					1
Update frequency	1					1
Compiler	1					1
Standard version	1					1
Data sources				18		18
Data transformations				18		18
Preprocessing steps			13	17		30
Dataset size				16		16
Known / potential biases			8	14		22
Procedure for collection				14		14
Types of data				13		13
Presence of PII			8	11		19
Version description				10		10
Dataset statistics				8		8

Table B.9: Fields to include in BOM (continued)

Field	SBOM C&A	Critical Projects	ML (AI2)	ML (D3)	CPS	Total
Data labelers				8		8
Data usage history				1		1
Known problems			7	1		8
Train / test split				1		1
Part numbers					3	3
Testing and QA data					2	2
Deployment information					1	1
Functionality					1	1
Place of manufacture					1	1
Packaging					1	1
Price					1	1
Quantity					1	1
Additional notes					1	1
Limitations					1	1
Description of training data			17			17
Description of validation and testing data			14			14
Data version			13			13
Optimizers, loss functions, etc			13			13
Model structure / architecture			12			12
Training environment			11			11
Model hyperparameters						0
Model parameters			10			10
Fairness issues			6			6
Validation procedures			1			1
User experiences			1			1
Loss function			1			1
Model initialization			1			1
Total participants	**43**		**20**	**20**	**6**	**89**

Table B.10: Deficiencies in SBOM standards identified by participants (by count)

Deficiency	**SBOM C&A**	**Critical Projects**	**Total**
Tooling	8		8
Inconsistency	4	2	6
Too many standards	3	4	7
Comprehension	3	1	4
Missing fields	3		3
Language support	2	3	5
Underspecification	2	1	3
Overspecification	2		2
Young technology	2		2
Ambiguous guidance	2	4	6
Use case support	2		2
Different expectations	1	1	2
Unavailable information	1		1
ML support	1	1	2
Sbom distribution	1	5	6
Checking sbom quality	1	1	2
Management cost	1		1
No deficiencies	1	1	2
Keeping updated	1		1
Difficult to authenticate	1	1	2
Low adoption		1	1
Usefulness unclear		1	1
Verifying SBOM		4	4
Total participants	**23**	**12**	**35**

Table B.11: Challenges identified by participants (by count)

Challenge	C&A (C7)	C&A (P9)	Critical (C6)	Critical (P8)	ML (C2)	ML (D8)	Total
Inconsistency	9				2	3	14
Poor quality SBOM	10	4	1				15
Missing tool features	3						3
Different standards	7						7
Verifiability	3				1	2	6
Upkeep requirements	1		1			3	5
Comprehension	2	1					3
Dynamic vulnerability data	1		1				2
Tool bugs	2						2
Tooling		8			1		9
Specification comprehension		3					3
Environment support		3		1			4
Legal requirements		1					1
Underspecification		1					1
Unavailable information		5		1		1	7
Implementation difficulty		2					2
Interoperability issues		1					1
Fast releases		2			1		3
Security risks		1			1	2	4
Versioning issues	1	1			1	2	5
Hard to get	1					1	2
No issues	1	2	1		1		5
Too many languages in project		2					2
Sbom too large		1			3	2	6
Sbom generation slow		1					1
Contribution issues		1					1
Separated from devs		1					1
Not backwards compatible		1					1
False positives		1					1
Number of Sboms		1					1
Cost		1					1
No response				1	1		2
Adoption					4	1	5
Witholding information					3		3
Time consuming					3		3
Use cases					2		2
Not useful					2	1	3
Documentation					2	1	3
Ensuring confidentiality					2		2
Domain specific knowledge					1		1
Distribution					1	1	2
Cross model					1		1
Too structured					1	1	2
Maintenance					1	1	2
IP and privacy						4	4
Regulation						1	1
Hashes						1	1
Representing labelers						1	1
Tracking training data						1	1
Total participants	**30**	**34**	**4**	**3**	**20**	**20**	**111**

Table B.12: Solutions proposed by participants (by count)

Solution	**SBOM C&A**	**Critical Projects**	**Total**
Support organizations	4	3	7
Higher quality sbom	4	2	6
Simpler tools	4	3	7
No deficiencies	3		3
Clarified standards	3	5	8
Automation	3		3
Generate accurate SBOM	2		2
More language support	2		2
Language specific tools	2	5	7
Report generation tools	2		2
Faster generation	2		2
Communication	2		2
Change tracking	1		1
Sbom database	1		1
Better libraries	1	2	3
Language agnostic tools	1	3	4
Show unknowns	1		1
Dynamic sboms	1		1
Let field mature	1		1
Patch propagation	1		1
Integrate security tools	1		1
Support for AI applications	1		1
Compatibility between tools	1		1
Sboms too large	1		1
Focus on existing tools	1		1
Include reachability graphs	1		1
Version compatibility matching	1		1
Intregate in package managers	1		1
No response		2	2
Total participants	**44**	**12**	**56**

Appendix C

Full Comparison with Related Works

Here I provide a more detailed, granular comparison with the most closely related works. Table C.1 provides a scope comparison for each of the studies, and Table C.2 provides metrics on the challenges discussed in and across all works. Coverage is defined as the percentage of challenges a work discusses compared to the total pool of identified challenges. Novel challenges are those that are unique to a work and not discussed in any other.

In Table C.3, column 3 indicates the unique SBOM-related issues/findings we identified in all publications. Columns 1 and 2 indicate the challenges we identified in our study. The issue/findings are grouped by challenge. Columns 4 - 7 are the existing SBOM studies, including ours. In these columns, we use **X** to indicate a study/work that has explicitly mentioned or discussed an issue/finding. In most cases, these were pulled directly from the groups findings, but in other instances, were mentioned as asides or intuitions. For each work, we also include where in the text the finding was mentioned. For Xia *et al.* [134], this maps to a portion of the results section. For Zahan *et al.* [137], to the appropriate challenge number. For the Linux Foundation report (Hendricks) [74], to relevant figures that show the challenge. The broad challenge IDs come from our work, so we only denote a specific location when it does not match the given ID. Blank cells mean an issue/finding is not mentioned or discussed at all in a publication. Where it was disputable whether or not a challenge was actually brought to light, we err in favor of the compared work. For

Table C.1: SBOM study methodology and scope

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 developers/researchers
Xia *et al.*'s [134]	Interviews to derive one survey	SBOMs & AIBOMs	Developers
Zahan *et al.*'s [137]	Grey literature review	SBOMs	-
Hendricks [74]	One survey	SBOMs	Software organizations

example, if AIBOM was mentioned, that would be enough for *C5: Lack of AI / ML tools.* Where it was disputable whether or not we covered a challenge, we err against our novelty and don't include the **X**.

Table C.2: Challenge coverage of related works

Work	Challenges Discussed	Coverage	Novel Challenges
Thesis	40	95.24%	12
Xia [134]	23	54.76%	2
Zahan [137]	18	42.86%	0
Hendrick [74]	8	19.05%	0
All	42	100%	-

Table C.3: Challenges discussed across contemporary works

Label	Challenge Category	Issue / Concept	Thesis	Xia [134]	Zahan [137]	Hendricks [74]
C1	Complexity of SBOM Specifications	Complexity of Specifications	X	X (RQ3-1)		
		Lack of educational resources	X	X (RQ3-3)	X (4)	X (Fig 27)
		Not extensible, missing use-cases	X	X (RQ3-1)	X (1)	
C2	Determining data fields to include in SBOM	Determining data fields to include in SBOM	X	X (RQ1-4)		X (Fig 23 / 26)
C3	Incompatibility between SBOM standards	Competing standards	X	X (RQ3-1)	X (2)	
		Inconsistency in tooling output	X	X (RQ2-3)	X (2)	
		Multiple representations within standard	X			
C4	Keeping SBOMs up to date	Managing SBOM changes over time (SBOM drift)	X			
		Upkeep requirements	X	X (RQ1-3)	X (1)	
		Dynamic / runtime BOM	X	X (Implications-1)	X (1)	
C5	Insufficient SBOM tooling	General tool insufficiency	X		X (1)	X (Fig 23 / 26)
		Lack of support of multi-language projects	X			
		Lack of AI / ML tools	X	X (RQ1-8)		
		Lack of CPS tools	X			
		Poor usability of tools		X (RQ2-3)		
		Lack of consumption-scenario-driven design		X (Finding 1)		
		Tools not enterprise ready	X	X (RQ2-3)		
C6	Inaccurate and incomplete SBOMs	Poor quality SBOM	X	X (RQ1-1)	X (1)	
		Reporting transitive dependencies	X		X (4)	X (SBOM Needs)
		Exploitability vs vulnerability	X	X (RQ1-7)	X (3)	
C7	Verifying SBOM accuracy and completeness	Verifying SBOM accuracy and completeness	X	X (RQ1-6)		
		Liability for incorrect SBOM	X		X(5)	X (Fig 23)
		SBOM integrity checking	X	X (RQ1-6)		
C8	Differences across ecosystems and communities	Supporting languages without package managers	X			
		Lack of and varying language support	X			
		Need for language specific tooling	X			
C9	SBOM completeness and data privacy trade-off	Protecting PII, proprietary information, etc. in AIBOM	X			
		Security and trade secrets	4.1.2	X (RQ1-5) / (RQ3-2)	X (5)	X (Fig 23)
C10	SBOMs for legacy packages and repositories	Making SBOM for legacy systems	X		X (3)	
		Retro-active SBOM creation for past versions	X			
C11	Inability to locate dependencies for SBOM	Locating dependencies removed from origin	X			
C12	Unclear SBOM direction	Unclear SBOM direction	X			X (Fig 23 / 26)
		Fear of minimal compliance document	X			
		Lack of adoption	4.1.1	X (RQ1-1)		
		Lack of incentives to generate SBOM	X	X (Finding 2)	X (4)	
		Uncertain value (work could outweigh benefits)	X	X (RQ1-1)	X (3)	X (Fig 23 / 26)
		SBOM generation is ahead of consumption	X	X (RQ2-2)		
		Effort intensive and time consuming SBOM generation	X		X (4)	
C13	Generating global software IDs	Lack of widely accepted unique package identifier	X		X (2)	
		Issues generating proper IDs	X			
C14	Versioning SBOM	Need for SBOM version control	X	X (RQ1-3)		
Other	SBOM Distribution	Distribution	4.1.3	X (RQ1-5)	X (2)	

Appendix D

Further Details on Response Filtering

Here we provide a more detailed explanation of how responses were removed from analysis.

Annotators examined the survey responses and independently flagged potentially invalid responses due to various reasons (explained below). They discussed the cases and reached a consensus on which responses to remove and the main reason for removal.

The 79 removed survey responses were:

- 1 response appeared to be generated by ChatGPT. It contained many verbose answers to survey questions, which often used unusual markup such as bullet points and Unicode characters. We manually asked ChatGPT the same survey questions, obtaining similar answers.
- 8 responses contained answers that were a verbatim copy of web content. We validated this by searching the answer text using Google.
- 32 responses contained nonsensical answers or answers that did not answer the survey questions. These responses contained multiple answers that did not make sense or used buzzwords or key terms in meaningless or incorrect ways.

For example, when asked to “Briefly describe the concept of SBOMs. What are they? What is their purpose?", we found vague, unhelpful, or meaningless answers such as: “Cyber Security Leader," “perfect the system," “development tool," and “c." One participant even answered: “I don’t know something you need to touch yourself."

When asked which data fields should be included in SBOMs, we got bogus answers such as: “the needed fields" or “bill of materials."

- In 13 responses, the participants repeated one word or phrase as the answer for every or most questions (*e.g.*, answers such as “yes," “no," and “n/a.")
- In 7 responses, the majority of the survey is left blank, despite being submitted. These participants skipped (nearly) all questions in the survey (by design, many questions were not required and could be skipped).
- In 18 responses, the participants completed the survey multiple times evidenced by a duplicated pair of email/IP addresses and by duplicated answers. Often these responses exhibited the other problems described above.

Most of these bogus responses were because, in our second wave of invitations, one of our industry contacts promoted the SBOM C&P survey in their network (via social media). Survey compensation, while seeking to attract participants also resulted in multiple bogus responses.

The remaining responses that continued to analysis passed all of our tests, including both the web search test and the ChatGPT test, leaving us with a high degree of confidence that the remaining responses are reliable.

Appendix E

Survey Questions

Here I provide the complete list of survey questions and their associated IDs. The questions are alphabetized and grouped by survey to facilitate efficient look up.

E.1 SBOM C&A

(C2) What do you use SBOMs for?

(C3) Please elaborate on the benefits from consuming SBOMs that you identified

(C4) How often do you use SBOMs during software development?

(C5) Do you process or analyze the SBOMs of the software dependencies that you use?

(C6) How do you process or analyze the SBOMs of your dependencies? What tools, if any, are involved in the process?

(C7) What issues have you faced when consuming SBOMs?

(E2) Software stakeholders understand the purposes / capabilities of SBOMs

(E3) Software stakeholders understand how to use SBOMs

(E4) Which SBOM formats do you teach about / compile resources for?

(E5) What types of resources are best to inform people about SBOMs?

(P2) At what point in the software development process should SBOMs be generated?

(P3) Why do you create SBOMs?

(P4) Please elaborate on the benefits from creating SBOMs that you identified

(P5) What tools do you use to assist the creation fo SBOMs, if any?

(P7) Does your organization have strategies for managing SBOM versions?

(P8) What strategies does your organization have for managing SBOM versions? Are there any specific tools involved?

(P9) What issues have you faced when creating SBOMs?

(Q1) How many years of experience in software development do you have?

(Q2) How would you describe your primary role?

(Q3) What is your highest level of education?

(Q4) Which programming languages have you most used in past projects?

(Q5) What types of systems have you developed?

(Q6) Do you primarily work on open source or closed source projects?

(Q8) Do you have a background in computer security?

(Q9) Do you have a background in software licensing?

(Q10) How often did you release or help release a new major version of an application / software over the past two years?

(S1) Briefly describe the concept of SBOMs. What are they? What is their purpose?

(S2) Which SBOM formats do you use?

(S3) Which data fields do you think should be included in SBOMs?

(S4) The use of SBOMs is critical in software development

(S5) What deficiencies currently exist in SBOM standards / specifications?

(S6) How can we address current deficiencies in SBOM standards / specifications?

(S7) Current SBOM tool support meets the needs of users

(S8) How can we address current deficiencies in SBOM tooling?

(S9) How would you prefer SBOMs be distributed?

(S10) Feel free to explain your preferences about how SBOMs should be distributed

(SC1) Vulnerabilities reported for software dependencies I use are important issues for my organization.

(SC2) Are you involved with writing, reviewing, or maintaining source code?

(SC3) I trust security scanners and static analysis tools to find vulnerabilities in my code.

(SC4) I trust that my dependencies are free from security vulnerabilities.

(SC5) Do you use a dependency tracking system?

(SC6) If you use a dependency tracking system, does it allow you to easily determine if your project is impacted by a discovered vulnerability?

(SC7) What dependency tracking system do you use, and how does it allow you to determine if your project is impacted by a discovered upstream vulnerability?

(SC8) How do you evaluate the security vulnerabilities of the dependencies you use?

(SC9) In the sofrware that you develop, how do you obtain the list of libraries that it uses?

(SM2) Where should standards / specifications regarding SBOM creation and usage originate from?

(SM3) How should SBOM standards / specifications be effectively communicated?

(T2) Which SBOM formats does your tool support?

(T3) What do you use SBOMs for?

(T4) Who is your tool primarily designed for?

(T5) In my industry, developers and other stakeholders are aware of the SBOM tools available to them

(T6) Which SBOM tools have you developed?

E.2 Critical Projects

(C1) What do your projects use SBOMs for?

(C2) Which of the following benefits have your projects observed from using SBOMs?

(C3) How often do your projects use SBOMs during software development?

(C4) Do your projects process or analyze the SBOMs of their software dependencies?

(C5) How do your projects process or analyze the SBOMs of their dependencies?

(C6) Which issues have your projects encountered in using SBOMs?

(C7) How do you prefer the SBOMs of other projects be shared with you?

(EP1) How do your projects publish / distribute your created SBOMs?

(FN1) Which of the following SBOM formats have you heard of?

(FN2) Have you used SBOMs in the past?

(FN3) Why do you not currently SBOMs?

(FS1) Which SBOM formats do your projects use?

(FS2) The use of SBOMs is critical in software development

(FS3) Current SBOM standards / specifications meet the needs of users and industry

(FS4) Which deficiencies have you or your projects encountered in current SBOM standards / specifications?

(FS5) How can we address the selected deficiencies in SBOM standards / specifications?

(FS6) Current SBOM tool support meets the needs of users

(FS7) What are some issues that you or your projects have encountered in SBOM tooling?

(FS8) How can we address current deficiencies in SBOM tooling?

(FS9) Do your projects have a process to verify whether SBOMs completely and correctly report all dependencies and their metadata?

(FS10) What process do your projects have to verify whether SBOMs completely and correctly report all dependencies and their metadata?

(ID) How are SBOMs used in your projects?

(IP1) Do your projects share their SBOMs externally?

(IP2) Why are your projects' SBOMs not shared externally?

(NU1) What tools and / or techniques do you use for dependency management in your software projects?

(NU2) Why do you prefer these tools and / or techniques over other methods?

(O1) Are you familiar with Operations Bills of Materials?

(O2) How would you describe OBOMs? What are their key features?

(O3) Do you use OBOMs in your projects?

(O4) What should OBOMs be used for?

(P1) Select all statements that apply to your projects

(P2) At what point in the software development process should SBOMs be generated?

(P3) Why do your projects create SBOMs?

(P4) What benefits have your projects observed from producing SBOMs?

(P5) What tools do your projects use to assist in the creation of SBOMs, if any?

(P6) Do your projects have any strategies for managing SBOM versions?

(P7) What strategies do your projects have for managing SBOM versions?

(P8) What issues have your projects encountered when producing SBOMs?

(Q1) How many years of experience in software development do you have?

(Q2) How would you describe your primary role?

(Q3) What is your highest level of education?

(Q4) Which programming languages have you most used in past projects?

(Q5) What types of systems have you developed?

(Q6) Do you primarily work on open source or closed source projects?

(Q7) For which domains have your developed applications?

(Q8) Do you have a background in computer security?

(Q9) Do you have a background in software licensing?

(Q10) How often did you release or help release a new major version over the past two years?

(Q11) Which countries are you or your organization based in?

(S1) Other well known / well established open source projects are producing SBOMs

(S2) Other well known / well established open source projects are consuming SBOMs

(S3) As an open source project, do you publish a list of your project's dependencies?

(S4) How do your projects obtain the list of dependencies?

(S5) Why do you projects not publish their dependencies?

E.3 Machine Learning

(AI1) Current SBOM formats adequately support machine/deep learning (ML/DL) systems.

(AI2) What data fields should be included in SBOMs to adequately describe a machine/deep learning (ML/DL) system?

(AI3) Can SBOMs of ML/DL systems (AIBOMs) follow the same current SBOM specifications and standards?

(AI4) Please describe a potential new solution for specifying SBOMs of ML/DL systems (AIBOMS).

(AI5) Are you aware of any tools which currently facilitate the automatic generation of AIBOMs?

(AI6) How would you prefer AIBOMs be distributed?

(AI7) How can we ensure that AIBOMs completely and correctly report all the dependencies of ML/DL systems?

(B1) Are you familiar with the concept of Software Bill of Materials?

(B2) Which of the following SBOM formats are you familiar with?

(B3) Are you aware of any existing SBOM standards / specifications for ML/DL systems (AIBOMs)?

(C1) What are some of the benefits you expect to see from using AIBOMs, if any?

(C2) What main challenges do you foresee in the creation and use of AIBOMs?

(C3) Do vulnerability databases (such as CVE) exist for ML/DL systems?

(C4) Vulnerability databases for ML/DL systems meet the security needs of developers and consumers

(C5) Do vulnerability databases (such as CVE) exist for datasets of ML/DL systems?

(C6) Vulnerability databases for datasets of ML/DL systems meet the security needs of developers and consumers

(C7) Given recent concerns regarding scraped public material being included in ma-

chine/deep learning datasets, how should AIBOMs address licensed material, if at all?

(D1) Datasets for ML/DL systems should come with a bill of materials

(D2) Are you aware of any existing standards for DataBOMs?

(D3) What data fields should be included in a DataBOM to adequately describe a dataset for a ML/DL system?

(D4) Are you aware of any tools that generate DataBOMs?

(D5) What do you think the relationship between AIBOMs and DataBOMs should be?

(D6) How should DataBOMs be shared / distributed?

(D7) What are some of the benefits you expect to see from using DataBOMs, if any?

(D8) What main challenges do you foresee in the creation and use of DataBOMs?

(D9) How can we ensure that DataBOMs completely and correctly report all the dependencies and/or components of datasets used by ML/DL systems?

(ID) How are SBOMs used in your projects?

(Q1) How many years of experience developing machine/deep learning systems do you have?

(Q2) How would you describe your primary role?

(Q3) What is your highest level of education?

(Q4) Which programming languages have you most used in past projects?

(Q5) Which machine / deep learning models do you use most frequently?

(Q6) Do you primarily work on open source or closed source projects?

(Q7) For which domains have your developed applications?

(Q8) Do you have a background in computer security?

(Q9) Do you have a background in software licensing?

(Q10) How often did you release or help release a new major version over the past two years?

(Q11) Which countries are you or your organization based in?

E.4 Cyber-Physical Systems

(G1) What are the challenges that people currently face in the supply chain of cyber-physical systems?

(G2) What is the role of open-source software in cyber-physical systems?

(HB1) Are you familiar with the concept of a Hardware Bill of Materials (HBOM)?

(HB2) Have you ever used HBOMs in a project?

(HB3) Which formats have you used for HBOMs?

(L1) SBOMs are a potential solution to problems that people face in the supply chain of cyber-physical systems

(L2) SBOMs can be used to effectively manage open-source components in cyber-physical systems

(L3) Current formats (e.g., SPDX or CycloneDX) convey all the information necessary to create inventories of physical components

(L4) For a cyber-physical system, the software and hardware manifests (i.e., inventories) should be separated into an SBOM and a HBOM (Hardware Bill of Materials).

(L5) SBOMs/HBOMs can be effectively used to demonstrate compliance with regulatory requirements and standards in cyber-physical systems.

(L6) Tool support exists for creating and processing SBOMs/HBOMs for cyber-physical systems.

(O1) What are the necessary data fields that an SBOM or HBOM must contain to accurately and sufficiently describe software/hardware components in cyber-physical systems?

(O2) What is the role of SBOMs/HBOMs in ensuring the traceability of components in cyber-physical systems?

(O3) How can the use of SBOMs/HBOMs impact risk assessment in cyber-physical systems?

(O4) How should SBOMs/HBOMs for cyber-physical systems be distributed? Rank by preference (by dragging the options).

(O5) (Optional) Do you have anything else you would like to note regarding SBOMs and HBOMs for cyber-physical systems?

(Q1) How many years of experience in software / hardware development do you have?

(Q2) How would you describe your primary role?

(Q3) What is your highest level of education?

(Q4) Which programming languages have you most used in past projects?

(Q5) What types of systems have you developed?

(Q6) Do you primarily work on open source or closed source projects?

(Q7) For which domains have your developed applications?

(Q8) Do you have a background in computer security?

(Q9) Do you have a background in software licensing?

(Q10) How often did you release or help release a new major version over the past two years?

(Q11) Which countries are you or your organization based in?

(SB1) Are you familiar with the concept of Software Bills of Material (SBOMs)?

E.5 Legal Practitioners

(AI2) In your opinion, would the use of complete and detailed DataBOMs be sufficient to cover the attribution requirements of open source software licenses?

(AI3) Do you anticipate DataBOMs being used with other large AI projects, like ChatGPT, that are trained on large corpora from the Internet?

(D1) Are you familiar with the concept of Software Bills of Materials (SBOMs)?

(D2) What do you consider to be the minimum adequate baseline component/dependency information for an SBOM?

(D3) In your estimation, how many SBOMs meet this minimum standard in practice?

(D4) Are you aware of any specific requirements that organizations must meet when cre-

ating SBOMs for software products that will be used in highly regulated industries, such as healthcare, finance, or defense? If so, please describe those requirements.

(L1) How do you think that an SBOM that includes information about the software and hardware components of such equipment could facilitate faster and more accurate approvals, if at all?

(L2) Are you aware of any disputes relating to an organization's creation or use of SBOMs?

(L3) If an SBOM is inaccurate or incomplete, what legal liability do you see for the entity that produced it?

(L4) What liability do you see for the creators of tools used to automatically generate SBOMs that are incomplete or inaccurate?

(L5) In your opinion, to what extent will the provision of an accurate and complete SBOM become a standard provision in software licenses and related contracts?

(L6) Should the creator of an SBOM be required to provide a certification that the SBOM is correct and complete? Please elaborate on your response.

(L7) What steps can organizations take to protect their proprietary information and trade secrets when creating and distributing SBOMs?

(L8) To what extent are organizations legally obligated to disclose any vulnerabilities or security issues they discover as a result of creating an SBOM to their customers, partners, or the public?

(Q1) How many years of legal experience do you have?

(Q2) Which country / countries are you and / or your organization based in?

(Q3) Please enter your email address. This will be used to contact you if you are identified as the recipient of the Amazon gift card.

(Q4) Would you be willing to participate in a follow-up interview based on your responses?

Appendix F

Image Credits

All symbols used to create Figure 3.1 were taken from www.flaticon.com [30] and www.freepik.com [31]. Table F.1 provides specific attribution details.

Table F.1: Attribution of symbols used

Symbol	**Creator**	**URL**
Industry Contacts	Eucalyp	https://www.flaticon.com/authors/eucalyp
Mailing Lists & Newsletters	surang	https://www.flaticon.com/authors/surang
GitHub Mining	srip	https://www.flaticon.com/authors/srip
Professional Network	Freepik	https://www.freepik.com/
SBOM C&A	Freepik	https://www.freepik.com/
Critical Projects	LAFS	https://www.flaticon.com/authors/lafs
Machine Learning	netscript	https://www.flaticon.com/authors/netscript
Cyber Physical Systems	smalllikeart	https://www.flaticon.com/authors/smalllikeart
Legal	monkik	https://www.flaticon.com/authors/monkik
Response Coding	srip	https://www.flaticon.com/authors/srip
Analysis	monkik	https://www.flaticon.com/authors/monkik
Interviews	netscript	https://www.flaticon.com/authors/netscript
RQ1	Smashicons	https://www.flaticon.com/authors/smashicons
RQ2	Uniconlabs	https://www.flaticon.com/authors/uniconlabs
RQ3	srip	https://www.flaticon.com/authors/srip

Bibliography

[1] CycloneDX History. https://cyclonedx.org/about/history/.

[2] GitOID. https://www.iana.org/assignments/uri-schemes/prov/gitoid.

[3] Openchain main mail list. https://lists.openchainproject.org/g/main.

[4] OWASP. https://owasp.org/.

[5] Software Heritage. https://www.softwareheritage.org/.

[6] SPDX Overview. https://spdx.dev/about/.

[7] Specification Overview. https://cyclonedx.org/specification/overview/.

[8] The Linux Foundation. https://www.linuxfoundation.org/.

[9] Using SPDX. https://spdx.dev/resources/use/.

[10] Spdx technical team use cases 2.0. https://wiki.spdx.org/view/Technical_Team/Use_Cases/2.0, 2013. Accessed: 2023-29-03.

[11] Cybersecurity supply chain risk management, 2016.

[12] Executive order 14028, 2021.

[13] What is a cve? https://www.redhat.com/en/topics/security/what-is-cve, 11 2021.

[14] Annex F External repository identifiers (Normative). https://spdx.github.io/spdx-spec/v2.3/external-repository-identifiers/#f42-gitoid, 2022.

[15] Common platform enumeration (cpe). https://csrc.nist.gov/Projects/Security-Content-Automation-Protocol/Specifications/cpe, 1 2022.

[16] Github rest api documentation. https://docs.github.com/en/rest?apiVersion=2022-11-28, 2022. Accessed: 2023-28-03.

[17] Sbom drift. https://docs.anchore.com/current/docs/sbom_management/sbom_drift/, 3 2022.

[18] Cyclonedx specifications, 2023.

[19] Harnessing GPT-4 so that all students benefit. A nonprofit approach for equal access. https://blog.khanacademy.org/harnessing-ai-so-that-all-students-benefit-a-nonprofit-approach-for-equal-access/, 3 2023.

[20] Introducing self-service sboms. https://github.blog/2023-03-28-introducing-self-service-sboms/, 3 2023. Accessed: 2023-29-03.

[21] ITI. https://www.itic.org/, 2023.

[22] purl-spec. https://github.com/package-url/purl-spec, 1 2023.

[23] Spdx specifications, 2023.

[24] Stripe and OpenAI collaborate to monetize OpenAI's flagship products. https://stripe.com/newsroom/news/stripe-and-openai, 3 2023.

[25] About the dependency graph. https://docs.github.com/en/code-security/supply-chain-security/understanding-your-software-supply-chain/about-the-dependency-graph, [n.d.]. Accessed: 2023-28-03.

[26] Anchore. https://anchore.com/platform/, [n.d.]. Accessed: 2023-29-03.

[27] boms away study. https://anonymous.4open.science/r/boms_away_study-2133/, [n.d.]. Accessed: 2023-29-03.

[28] Cc0 1.0 universal (cc0 1.0) public domain dedication. https://creativecommons.org/publicdomain/zero/1.0/, [n.d.]. Accessed: 2023-29-03.

[29] Data version control. https://dvc.org/, [n.d.]. Accessed: 2023-29-03.

[30] Flaticon. https://www.flaticon.com/, [n.d.].

[31] Freepik. https://www.freepik.com/, [n.d.].

[32] The mit license. https://opensource.org/license/mit/, [n.d.].

[33] mlflow. https://mlflow.org/, [n.d.]. Accessed: 2023-29-03.

[34] Qualtrics. https://www.qualtrics.com/, [n.d.]. Accessed: 2023-28-03.

[35] Scancode. https://www.nexb.com/scancode/, [n.d.]. Accessed: 2023-29-03.

[36] Spdx object property: datalicense. https://spdx.org/rdf/spdx-terms-v2.1/objectproperties/dataLicense___1140128580.html, [n.d.]. Accessed: 2023-29-03.

[37] spdx@lists.spdx.org. https://lists.spdx.org/g/spdx, [n.d.]. Accessed: 2023-28-03.

[38] 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, [n.d.].

[39] Amy Nelson, Jiewen Yao, Vincent Zimmer. Traceable firmware bill of materials overview. https://uefi.org/sites/default/files/resources/Traceable%20Firmware%20Bill%20of%20Materials%20-%2020211207%20-%20007.pdf, 12 2021.

[40] Andrei Costin. Securing your iot device with fboms from devastating cyberattacks. https://euhubs4data.eu/blog/securing-iot-device-with-fboms/, 4 2022.

[41] Arushi Arora, Virginia Wright, and Christina Garman. Strengthening the security of operational technology: Understanding contemporary bill of materials. *JCIP The Journal of Critical Infrastructure Policy*, 3(1):111, 2022.

[42] Aadesh Bagmar, Josiah Wedgwood, Dave Levin, and Jim Purtilo. I know what you imported last summer: A study of security threats in thepython ecosystem. *arXiv preprint arXiv:2102.06301*, 2021.

[43] Musard Balliu, Benoit Baudry, Sofia Bobadilla, Mathias Ekstedt, Martin Monperrus, Javier Ron, Aman Sharma, Gabriel Skoglund, César Soto-Valero, and Martin Wittlinger. Challenges of producing software bill of materials for java. *arXiv preprint arXiv:2303.11102*, 2023.

[44] Sebastian Baltes and Stephan Diehl. Worse than spam: Issues in sampling software developers. In *Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–6, 2016.

[45] Iain Barclay, Alun Preece, Ian Taylor, Swapna Krishnakumar Radha, and Jarek Nabrzyski. Providing assurance and scrutability on shared data and

machine learning models with verifiable credentials. *Concurrency and Computation: Practice and Experience*, page e6997, 2022.

[46] Iain Barclay, Alun Preece, Ian Taylor, and Dinesh Verma. Towards traceability in data ecosystems using a bill of materials model. *arXiv preprint arXiv:1904.04253*, 2019.

[47] Emily M Bender and Batya Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018.

[48] "Bill Bensing". History of the Software Bill of Material (SBOM). `https://billbensing.com/software-supply-chain/history-software-bill-of-material-sbom/`, 7 2022.

[49] Brian Ka Chan. Artificial intelligence bill of materials (aibom). `https://minddata.org/bill-of-artificial-intelligence-materials-boaim-Brian-Ka-Chan-AI`, 11 2017.

[50] Seth Carmody, Andrea Coravos, Ginny Fahs, Audra Hatch, Janine Medina, Beau Woods, and Joshua Corman. Building resilient medical technology supply chains with a software bill of materials. *NPJ Digital Medicine*, 4(1):34, 2021.

[51] Peter J Caven, Shakthidhar Reddy Gopavaram, and L Jean Camp. Integrating human intelligence to bypass information asymmetry in procurement decision-making. In *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, pages 687–692. IEEE, 2022.

[52] Kai Chen, Xueqiang Wang, Yi Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Bin Ma, Aohui Wang, Yingjun Zhang, and Wei Zou. Following devil's footprints: Cross-platform analysis of potentially harmful libraries on

android and ios. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 357–376. IEEE, 2016.

[53] Catalin Cimpanu. Ten malicious libraries found on pypi - python package index. https://www.bleepingcomputer.com/news/security/ten-malicious-libraries-found-on-pypi-python-package-index/, 9 2017. Accessed: 2023-27-03.

[54] Cloud Security Alliance. Saas governance best practices for cloud customers. https://cloudsecurityalliance.org/artifacts/saas-governance-best-practices-for-cloud-customers/, 10 2022.

[55] CycloneDX. https://cyclonedx.org/.

[56] CycloneDX. Hardware bill of materials (hbom). https://github.com/CycloneDX/bom-examples/tree/master/HBOM, 7 2022.

[57] CycloneDX. Operations bill of materials (obom). https://github.com/CycloneDX/bom-examples/tree/master/OBOM, 1 2022.

[58] CycloneDX. Software-as-a-service bom (saasbom). https://github.com/CycloneDX/bom-examples/tree/master/SaaSBOM, 1 2022.

[59] CycloneDX. Capabilities. https://cyclonedx.org/capabilities/, [n.d.].

[60] David Waltermire, Brant A. Cheikes, Larry Feldman, Greg Witte. Guidelines for the creation of interoperable software identification (swid) tags. https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8060.pdf, 4 2016.

[61] Massimiliano Di Penta, Daniel M. Germán, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 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*, pages 145–154, 2010.

[62] Shannon Leigh Eggers, Drew Christensen, Tori Brooke Simon, Baleigh Rae Morgan, and Ethan S Bauer. Towards software bill of materials in the nuclear industry. Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2022.

[63] Eliot Beer. Firmware security in the spotlight after novel ransomware attacks. `https://thestack.technology/firmware-attacks-focus/`, 6 2022.

[64] William Enck and Laurie Williams. Top five challenges in software supply chain security: Observations from 30 industry and government organizations. *IEEE Security Privacy*, 20(2):96–100, 2022.

[65] Hugging Face. Dataset cards. `https://huggingface.co/docs/hub/datasets-cards`, [n.d.]. Accessed: 2023-29-03.

[66] FOSSA Inc. A Practical Guide to CycloneDX. `https://fossa.com/learn/cyclonedx`.

[67] FOSSA Inc. CycloneDX vs SPDX. `https://www.youtube.com/watch?v=IQledp8WccU`, 2 2023.

[68] GR Gangadharan, Vincenzo D'Andrea, Stefano De Paoli, and Michael Weiss. Managing license compliance in free and open source software development. *Information Systems Frontiers*, 14:143–154, 2012.

[69] GAO. Federal agencies need to address aging legacy systems. `https://www.gao.gov/assets/files.gao.gov/assets/gao-16-696t.pdf`, 5 2016.

[70] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[71] Google. Understanding the Impact of Apache Log4j Vulnerability. https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html, 12 2021.

[72] Robert M. Groves, Floyd J. Jr. Fowler, Mick P. Couyper, James M. Lepkowski, Eleanor Singer, and Roger Tourangeau. *Survey Methodology, 2nd edition*. Wiley, 2009.

[73] GuardRails. 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/, 2 2023. Accessed: 2023-29-03.

[74] Stephen Hendrick. Software bill of materials (sbom) and cybersecurity readiness, 2022.

[75] Henk Birkholz, Jessica Fitzgerald-McKay, Charles Schmidt, David Waltermire. Concise software identification tags. https://www.ietf.org/archive/id/draft-ietf-sacm-coswid-19.html, 10 2021.

[76] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. *arXiv preprint arXiv:1805.03677*, 2018.

[77] ISO. Iso/iec 5962:2021 information technology — spdx specification v2.2.1. https://www.iso.org/standard/81870.html, 2021.

[78] ISO. Iso/iec 19770-2:2015. https://www.iso.org/standard/65666.html, 3 2023.

[79] Laman Jalilova. Report on consolidation of the cern accelerator build infrastructure. https://cds.cern.ch/record/2778929/files/Laman_Jalilova_CERN_Report.pdf, 8 2021.

[80] Andrew Jamieson. Quantifying complexity: The challenges of supply chain security. https://www.eetimes.com/quantifying-complexity-the-challenges-of-supply-chain-security/, 10 2020. Accessed: March 26, 2023.

[81] Wenxin Jiang, Nicholas Synovic, Matt Hyatt, Taylor R Schorlemmer, Rohan Sethi, Yung-Hsiang Lu, George K Thiruvathukal, and James C Davis. An empirical study of pre-trained model reuse in the hugging face deep learning model registry. *arXiv preprint arXiv:2303.02552*, 2023.

[82] Wenxin Jiang, Nicholas Synovic, Rohan Sethi, Aryan Indarapu, Matt Hyatt, Taylor R Schorlemmer, George K Thiruvathukal, and James C Davis. 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*, pages 105–114, 2022.

[83] John P. Mello Jr. Sboms in the saas era: 5 reasons why you should consider a saasbom. https://www.reversinglabs.com/blog/5-reasons-why-you-need-a-saasbom, 11 2022.

[84] Josh Bressers. Fast and furious: Doubling down on sbom drift. https://thenewstack.io/fast-and-furious-doubling-down-on-sbom-drift/, 11 2022.

[85] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research part 2: designing a survey. *ACM SIGSOFT Software Engineering Notes*, 27(1):18–20, 2002.

[86] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research: part 3: constructing a survey instrument. *ACM SIGSOFT Software Engineering Notes*, 27(2):20–24, 2002.

[87] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research part 4: questionnaire evaluation. *ACM SIGSOFT Software Engineering Notes*, 27(3):20–23, 2002.

[88] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research: part 5: populations and samples. *ACM SIGSOFT Software Engineering Notes*, 27(5):17–20, 2002.

[89] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research part 6: data analysis. *ACM SIGSOFT Software Engineering Notes*, 28(2):24–27, 2003.

[90] Ravie Lakshmanan. 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.

[91] Ravie Lakshmanan. Extremely critical log4j vulnerability leaves much of the internet at risk. `https://thehackernews.com/2021/12/extremely-critical-log4j-vulnerability.html`, 12 2021. Accessed: 2022-05-12.

[92] Ravie Lakshmanan. Malicious npm package caught mimicking material tailwind css package. `https://thehackernews.com/2022/09/malicious-npm-package-caught-mimicking.html`, 9 2022. Accessed: 2023-27-03.

[93] Ravie Lakshmanan. Multiple backdoored python libraries caught stealing aws secrets and keys. `https://thehackernews.com/2022/06/multiple-backdoored-python-libraries.html`, 6 2022. Accessed: 2023-27-03.

[94] Ravie Lakshmanan. Researchers uncover pypi package hiding malicious code behind image file. https://thehackernews.com/2022/11/researchers-uncover-pypi-package-hiding.html, 11 2022. Accessed: 2023-27-03.

[95] Lawrence Abrams. YouTube-dl removed from GitHub after RIAA DMCA notice. https://www.bleepingcomputer.com/news/software/youtube-dl-removed-from-github-after-riaa-dmca-notice/, 10 2020.

[96] Genpei Liang, Xiangyu Zhou, Qingyu Wang, Yutong Du, and Cheng Huang. Malicious packages lurking in user-friendly python package index. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 606–613. IEEE, 2021.

[97] Everist Limaj, Edward Bernroider, and Maria Ivanova. Facing legacy information system modernization in scaling agility in the banking industry: Preliminary insights on strategies and non-technical barriers. 2020.

[98] Lu Lin et al. Generating software bill of material for vulnerability management and license compliance. 2023.

[99] Robert Alan Martin. Visibility & control: addressing supply chain challenges to trustworthy software-enabled things. In *2020 IEEE Systems Security Symposium (SSS)*, pages 1–4. IEEE, 2020.

[100] Jeffrey G. Miller and Linda G. Sprague. Behind the growth in materials requirements planning. *Harvard Business Review*, 1975.

[101] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[102] MITRE. Panel discussion: Making swid tags successful in the marketplace. https://csrc.nist.gov/CSRC/media/Presentations/Panel-Discussion-Making-SWID-Tags-Successful-in-t/images-media/day1_security-automation_200-250.pdf, 9 2015.

[103] NIST. CVE-2021-44228 Detail. https://nvd.nist.gov/vuln/detail/CVE-2021-44228, 12 2021.

[104] NTIA. Framing software component transparency: Establishing a common software bill of material (sbom). https://ntia.gov/files/ntia/publications/framingsbom_20191112.pdf, 2019.

[105] NTIA. Roles and Benefits for SBOM Across the Supply Chain. https://ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf, 2019.

[106] NTIA. Sbom at a glance. https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_apr2021.pdf, 2021.

[107] NTIA. Sbom myths vs. facts, 2021. Accessed: 2023-29-03.

[108] NTIA. SBOM Tool Classification Taxonomy. https://ntia.gov/files/ntia/publications/ntia_sbom_tooling_taxonomy-2021mar30.pdf, 2021.

[109] NTIA. Sharing and Exchanging SBOMs. https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf, 2021.

[110] NTIA. Software bill of materials elements and considerations. https://ntia.gov/sites/default/files/publications/uscc_-_2021.06.17_0.pdf, 2021.

[111] NTIA. Survey of existing sbom formats and standards, 2021.

[112] Phil Odence. Why you should use spdx for security. https://www.linux.com/featured/why-you-should-use-spdx-for-security/, 1 2023.

[113] Marc Ohm, Henrik Plate, Arnold Sykosch, and Michael Meier. Backstabber's knife collection: A review of open source software supply chain attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 17th International Conference, DIMVA 2020, Lisbon, Portugal, June 24–26, 2020, Proceedings 17*, pages 23–43. Springer, 2020.

[114] OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt, 11 2022. Accessed: 2023-29-03.

[115] OpenSSF. Securing critical projects workgroup: List of projects identified as 'critical'. https://docs.google.com/spreadsheets/d/1ONZ4qeMq8xmeCHX03lIgIYE4MEXVfVL6oj05lbuXTDM/edit#gid=1024997528, 2022.

[116] Sean Peisert, Bruce Schneier, Hamed Okhravi, Fabio Massacci, Terry Benzel, Carl Landwehr, Mohammad Mannan, Jelena Mirkovic, Atul Prakash, and James Bret Michael. Perspectives on the solarwinds incident. *IEEE Security & Privacy*, 19(2):7–13, 2021.

[117] Shari Lawrence Pfleeger and Barbara A. Kitchenham. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes*, 26(6):16–18, 2001.

[118] Martin Pratoussy. Establishment of a new workflow to manage software vulnerabilities. https://cds.cern.ch/record/2826626/files/Report-PRATOUSSY_Martin.pdf, 9 2022.

[119] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

[120] Rezilion. Dynamic sbom: A comprehensive guide. `https://www.rezilion.com/blog/dynamic-sbom-a-comprehensive-guide/`, 3 2022.

[121] Dirk Riehle and Nikolay Harutyunyan. 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*, pages 83–95. Springer, 2019.

[122] Guillaume Rousseau, Roberto Di Cosmo, and Stefano Zacchiroli. Software provenance tracking at the scale of public source code. *Empirical Software Engineering*, 25:2930–2959, 2020.

[123] PS Rusk. The role of the bill of material in manufacturing systems. *Engineering Costs and Production Economics*, 19(1-3):205–211, 1990.

[124] Ryan Naraine. Big tech vendors object to us gov sbom mandate. `https://www.securityweek.com/big-tech-vendors-object-us-gov-sbom-mandate/`, 12 2022.

[125] Adriana Sejfia and Max Schäfer. Practical automated detection of malicious npm packages. *arXiv preprint arXiv:2202.13953*, 2022.

[126] Neil Sheppard. Sboms (software bill of materials): Why do they matter?, 3 2023.

[127] Donna Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[128] Xin Tan, Kai Gao, Minghui Zhou, and Li Zhang. An exploratory study of deep learning supply chain. In *Proceedings of the 44th International Conference on Software Engineering*, pages 86–98, 2022.

[129] Wei Tang, Zhengzi Xu, Chengwei Liu, Jiahui Wu, Shouguo Yang, Yi Li, Ping Luo, and Yang Liu. Towards understanding third-party library dependency in c/c++ ecosystem. In *in ASE'22*, pages 1–12, 2022.

[130] Ann R. Thryft. The challenges of securing the open source supply chain.

[131] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 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, editors, pages 511–522. ACM, 2018.

[132] Christopher Vendome, Gabriele Bavota, Massimiliano Di Penta, Mario Linares-Vásquez, Daniel German, and Denys Poshyvanyk. License usage and changes: a large-scale study on github. *Emp. Soft. Eng.*, 22:1537–1577, 2017.

[133] Christopher Vendome, Mario Linares-Vásquez, Gabriele Bavota, Massimiliano Di Penta, Daniel German, and Denys Poshyvanyk. License usage and changes: a large-scale study of java projects on github. In *2015 IEEE 23rd International Conference on Program Comprehension*, pages 218–228. IEEE, 2015.

[134] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. An empirical study on software bill of materials: Where we stand and the road ahead. *arXiv preprint arXiv:2301.05362*, 2023.

[135] Henry Young. SBOMs: Considerable Progress, But Not Yet Ready for Codification. https://techpost.bsa.org/2022/08/31/sboms-considerable-progress-but-not-yet-ready-for-codification/.

[136] Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams. Software bills of materials are required. are we there yet? *IEEE Security & Privacy*, 21(2):82–88, 2023.

[137] Nusrat Zahan, Laurie Williams, Thomas Zimmermann, Patrice Godefroid, Brendan Murphy, and Chandra Maddila. What are weak links in the npm supply chain? *arXiv preprint arXiv:2112.10165*, 2021.