Bachelor of Science in Software Engineering
May 2023

Understanding the Software Bill Of Material for supply-chain management in Open Source projects

Veronica Axelsson & Frida Larsson

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Software Engineering. The thesis is equivalent to Weeks weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Veronica Axelsson
E-mail: veax20@student.bth.se, veronica.axelsson98@gmail.com

Frida Larsson
E-mail: frls19@student.bth.se, frida.l.larsson@outlook.com

University advisor:
Assistant Professor Davide Fucci
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

There has been an increase in the discussion about Software Bills of Material (SBOM) in the last few years, following a number of big-scale supply-chain attacks and vulnerabilities discovered in Open Source third-party packages. However, there is a lot to be done before the software community as a whole can fully reap the benefits SBOMs are claimed to provide.

The objective of this thesis is to investigate how far the Open Source software (OSS) community has come in adopting SBOMs, and how the existing SBOMs evolve, focusing on the Software Package Data Exchange (SPDX) format. For the purpose of this investigation an archival study was conducted, looking for SBOMs in OSS projects on GitHub and analyzing their content and evolution. This is one of the first large-scale searches for SBOMs in OSS projects, with the objective to research the practice of SBOM.

Only a fraction of the repositories that were inspected contained a SBOM, and most of them were found in Go projects. The SBOMs could be found in the source code of the repository, but the majority were found amongst the assets in the releases. Overall, the SBOMs were updated frequently using the latest SPDX format, and most stayed consistent with the quality of the content over time.

Keywords: SBOM, Software Bill of Material, SPDX, supply-chain management

Acknowledgments

We would like to express our gratitude to Davide Fucci for inspiring this thesis by introducing us to the subject of SBOMs. We would also like to thank him for the great support and feedback he provided us with throughout this research.

Further, we would like to thank John Speed Mayers, who created the Github repository Chainguard-dev/bom-shelter. This helped us immensely by providing information about finding SBOMs using Sourcegraph.

Contents

Chapter 1

Introduction

1.1 Background

The use of open source software (OSS) has increased steadily for decades, and today it is used everywhere by both governments and companies. OSS is published under an open license that provides the opportunity for anybody to examine, use, modify, and build upon the software, customizing it to their own needs. This is both cost-effective and greatly beneficial for innovation, but can also leave software vulnerable. In 2021, a vulnerability in the OSS Log4J was detected. This software was widely used by businesses and governments alike, leading to millions of devices being affected. Many software providers were not aware if or where Log4J was used in their code. When new versions of Log4J with security patches were released, not knowing where it was implemented made it a lot harder to fix the issue in the dependent software, increasing the effects of the vulnerability [1].

In the case of Log4J, the vulnerability was caused by a bug that was not created by a malicious actor [1]. However, these kinds of vulnerabilities can also be introduced by an attacker, in what is known as a supply-chain attack. A supply-chain attack happens when the attacker gains access to a third-party vendor code (e.g., in a GitHub repository) and replaces part of the source code or hides malware in it. The malicious code becomes a part of this component that other software depends on. This could result in a large number of potential victims even though just one software was originally targeted [2]. Apart from the vulnerability found in Log4J, another event that made a great impact on the software community was the attack on SolarWinds in 2020. SolarWinds provides system management tools to more than 30 000 organizations all over the world, so in attacking SolarWinds the attackers were able to also access their clients, including companies such as Microsoft, Intel, Cisco, as well as multiple American government departments [3]. In Sweden, the supermarket chain Coop was subject to a supply-chain attack in 2021, which caused the payment terminals to shut down. The reason was an attack on Kaseya VSA, a company that provides solutions to monitor Managed Service Providers. Kaseya VSA was used by Visma Esscom, which in turn Coop used in

their checkout system. The attack affected 60 of Kaseya VSA's customers, and since these clients in turn provide services to their own customers, the total number of affected organizations were about 1500 [4]. In the aftermath of the attack on SolarWinds, the Biden Administration introduced an executive order in May of 2021, calling for secure code practices when working with the American government. One of the requirements stated in the executive order is the use of Software Bill of Materials (SBOMs) [5]. A SBOM is a record of all the third-party components that make up the software, making it easy to find, if, where, and how OSS code is used. Therefore SBOMs are an effective way to mitigate supply chain attacks [1].

These recent attacks and the vulnerability found in Log4J have led to an increased discussion about measures against supply-chain attacks and the role that SBOMs can play in this. OSS are common targets for supply-chain attacks and most products depend on such software, in turn making them vulnerable [5]. With this in mind, it is interesting to study how OSS projects manage their SBOMs today. More specifically this study first takes a closer look at SBOMs in the NodeJS ecosystem, since according to a survey done by Statista, it was the most used web technology in 2022, with more than 47% of developers using NodeJS [6]. The study then broadens its perspective, not limiting the search to the NodeJS ecosystem, in an effort to find as many SBOMs as possible. Both searches were done using GitHub as a source since, according to the 2022 Stack Overflow Developer Survey, it is the most popular version control platform [7].

There are two common SBOM formats, Software Package Data Exchange (SPDX) and CycloneDX. SPDX is a project supported by the Linux foundation that dates back to 2010 while CycloneDX was created by OWASP[1] in 2017. SPDX is, apart from having existed the longest, the only SBOM format that is a public standard (ISO/IEC 5962.2021)[2]. Therefore, the SPDX format will be the focus of this study [8].

[1] https://owasp.org/
[2] https://www.iso.org/standard/81870.html

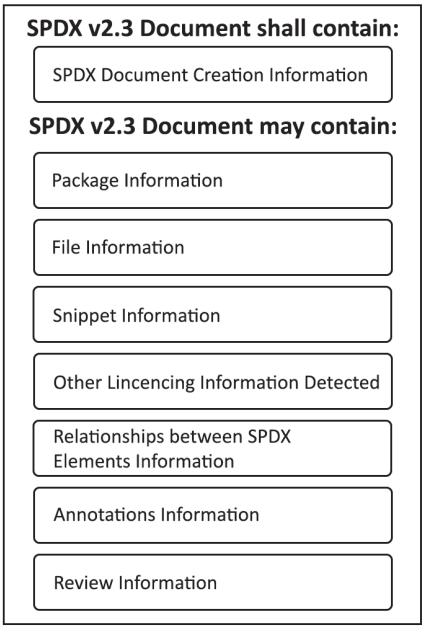

Figure 1.1: SPDX document sections. [9]

An SPDX document is made up of one or more sections, each section containing information in the form of fields. As can be seen in Figure 1.1 there is one mandatory section, SPDX Document Creation Information. This section contains seven mandatory and four optional fields. The mandatory fields are SPDX version field, Data license field, SPDX identifier field, Document name field, Creator field, Created field and SPDX document namespace field. The optional fields are External document references field, License list version field, Creator comment field and Document comment field. Apart from the mandatory section, there are seven optional sections which in turn have both mandatory and optional fields. These optional sections can be found in Figure 1.1 [9].

1.2 Motivation and Value

Since OSS projects are common targets for supply-chain attacks and are widely used in all types of software, it is of interest to investigate how and if OSS projects manage their SBOMs. This research will look into how common the occurrence of SBOMs in the SPDX format is in OSS projects. SBOMs can help to combat supply-chain attacks and as stated in the introduction, the discussion around this topic has increased due to the large-scale supply-chain attacks and the introduction of Biden's executive order. This research will give an insight into to what extent the OSS community actually utilizes SBOMs. It can also lay the ground for future

research, looking at the evolution of SBOMs and their occurrence.

Further, we will examine what data the SBOMs contain and if they include more or fewer fields than stated in the SPDX format. This will help determine the importance of different data fields and to aid further development of the SPDX format.

This research will investigate how the SBOMs evolve over time. SBOMs with the SPDX format have existed since 2010, and there are multiple tools that can be used to create them. However, there is a lack of ways and standards on how they are to be properly handled and updated. Zahan et al. [10] point out that the discussion around SBOMs is usually centered around the content of the SBOMs, and that there is a lack of focus on workflow automation and how the data is shared. Looking at when and if the SBOMs are updated and whether they change over time, as well as how they are shared and what tools are popular for creating them, could help in developing better practices that can benefit the OSS community. From the viewpoint of companies including OSS in their products, how the SBOMs evolve is of great importance for credibility and trust.

1.3 Scope

We initially limit the scope to the top 1000 downloaded NPM packages with associated GitHub repositories as NodeJS is the most popular technology used in web development [6], and NPM is the most popular package manager for NodeJS OSS libraries [11, 12]. To get a better quantitative overview of the current state of SBOM, the scope was expanded to GitHub repositories available on Sourcegraph [13]. The SBOM format investigated in this study will be SPDX. This research paper will not include results from SBOMs falling outside these criteria.

1.4 Outline

In the software community, the discussion about SBOMs is centered around what the SBOM file should and can include. This thesis wants to lift the topic of how SBOMs are created and then evolve, which is a relatively unexplored area. It examines a large dataset of SBOM files to increase the understanding of the current state of practise of SBOM. To find a good foundation for the study, we conduct a search for relevant articles and related work and perform an archival study to answer the research questions. We collect SBOMs by mining GitHub for SPDX files found in the releases and repositories of OSS projects. Data relevant to answer the research question is extracted from the SBOMs, using already existing tools and ad-hoc scripts.

Chapter 2

Related Work

The Linux Foundation conducted a study investigating SBOM readiness following the Biden Administrations executive order. 412 organizations from around the world answered a survey about their views on SBOM and other security concerns. 98% of the participating organizations use OSS in their products and 60% view SBOMs in OSS equally as important as in proprietary software while 29% view it as more important. 47% of the organizations used SBOMs in 2021 and the study estimates that the number of organizations using SBOMs will increase to 78% in 2022 and to 88% in 2023. The top three expected benefits of producing SBOMs were found to be that it aid developers in understanding dependencies, helps with vulnerability monitoring, and facilitate in complying with license obligations. Regarding the benefits of consuming SBOMs, the top three expected benefits were that SBOMs provide information about components that better supports compliance and reporting requirements, that SBOMs contribute to more informed risk-based decision-making and that it helps the organization to understand if they are at risk as new component vulnerabilities are discovered. Considering concerns for both producing and consuming SBOMs the top concern in both categories was that it is unclear whether the industry is committed to requiring SBOMs. For producing SBOMs the second biggest concern was whether there is a industry consensus on what a SBOM should contain [5].

Zahan et al. [10] performed a grey literature study to get a better understanding of the view on SBOM benefits and challenges. The article points out that only 18% of the organizations in the Linux Foundations SBOM readiness survey use SBOMs over nearly all their business segments, which leads to the question of why this is the case. The benefits highlighted were that it can help with dependency, vulnerability, risk, and license management, as well as transparency leading to better-informed purchasing decisions. Some of the challenges addressed were that most tools are suited for generating SBOMs for single components and not for a product as a whole. The tools also depend on guessing and information from top-level components when gathering package information. There is a lack of automation for detecting when components are modified, which is needed for the

SBOMs to be accurately updated. The article also points out that different clients require different data depending on use cases. It goes on to point out the lack of discussion around data exchange, workflow automation, and implementation when talking about SBOMs, and that it is mostly the content of the SBOMs that are in focus. The article concludes that industry standards and practices need to be further developed before a large-scale SBOM adaptation can take place.

In their article, Xia et al. [14] investigate the current state of SBOM practice and SBOM tooling support, and what the main concerns for SBOMs are. They gathered data from 17 interviews and from a survey with 65 respondents worldwide. They found that the majority of the respondents believe that SBOM distribution and generation require further regulations and more developed mechanisms. The results found in the study were compared to the Linux Foundation SBOM readiness report and found that the adoption of SBOMs is not as optimistic, with most existing third-party components or software not having a SBOM. This especially adheres to OSS projects and a better incentive for producing SBOMs is needed. The article also states that in the cases that SBOMs exists, the generation of them is often belated and not dynamic.

Carmody et al. [15] show how the value of SBOMs is starting to be recognized in the industry and that SBOMs will help in improving the public's trust in connected technologies. According to the article, the use of SBOMs is very likely to increase over the next few years, especially in life-critical software such as healthcare technologies.

A few articles propose different ways of aiding SBOM production. Nadgowda et al. [16] look at DevSecOps in micro-service applications. The study surveys GitHub repositories to find security gaps and challenges in the pipeline and then goes on to propose a single comprehensive DevSecOps pipeline, which includes a discovery engine that produces complete and accurate SBOMs. In another article, Nadgowda et al. [17] propose a design for a supply chain security framework that includes a new model for handling SBOMs, the SBOM-ctl. It manages all SBOM operations and allows for SBOMs to be treated not as document management, but rather as a distributed process. Martin [18] explores different use cases for SBOMs in an effort to understand what information need to be included and goes on to discuss how this data should be formatted.

Gandhi et al. [19] take a closer look at how the SPDX specification is used to guide improvement to OSS risk management routines in organizations in the SPDX community and how organizations contribute to the SPDX specification. One finding was that some organizations implemented the whole SPDX standard, while some only partly used it and others deemed it too complex. It also presses that the SPDX community has a big impact on the further development of the standard so that it adheres to the needs of its consumers.

Chapter 3

Method

3.1 Research questions

The focus of this study is on how SBOMs in the SPDX format are managed in OSS projects. In particular, it focuses on gaining a better understanding of how the SBOMs are created, what they contain, and how they evolve. The results of this research are the basis to improve the use of SBOMs.

- **RQ1.** How common is the occurrence of SBOMs in OSS projects hosted on GitHub?

  Supply chain attacks are a big threat against OSS projects and all software depending on them. A way of mitigating these attacks is the use of SBOMs, making it interesting to see how common the use of SBOMs in OSS projects hosted on GitHub is since GitHub is the most used version control platform.

- **RQ2.** To what extent do SBOMs in projects hosted on GitHub follow the SPDX standard?

  Standards for SBOM formats exist, the oldest and only one that is an international open standard being SPDX. Investigating how well the standard is followed and what information the SBOMs contain, can provide insights into what information is used and valued by OSS developers and consumers.

- **RQ3.** How do SBOMs evolve in OSS projects hosted on GitHub?

  SBOM files are not supposed to be static, but should evolve over time to be up to date with the changes in the project's software dependencies, as well as with changes in the SPDX standard. There are no standard practices for handling and updating SBOMs, therefore it would be beneficial for providers and consumers of SBOMs to learn more about what tools are

popular for creating SBOMs, how the data is shared, and to what extent they are kept up to date.

3.2 Literature review

To create a background for the study and to gain a better understanding of the subject, a literature review was conducted. The review looked into how SBOMs can be created, managed, and formatted. To motivate the importance of this study, supply-chain attacks and vulnerabilities in OSS projects were also investigated. To gather relevant information and material a literature search was conducted using Scopus and Google Scholar. The following search strings were used on Scopus:

- TITLE-ABS-KEY (spdx) AND PUBYEAR $>$ 2009 AND PUBYEAR $<$ 2024 AND (LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (LANGUAGE , "English"))
- sbom AND PUBYEAR $>$ 2009 AND PUBYEAR $<$ 2024 AND (LIMIT-TO (EXACTKEYWORD , "Supply Chains")) AND (LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (LANGUAGE , "English")
- TITLE-ABS-KEY ("software bill of material") AND PUBYEAR $>$ 2009 AND PUBYEAR $<$ 2024 AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI"))
- TITLE-ABS-KEY (software AND bill-of-material) AND PUBYEAR $>$ 2009 AND (LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (EXACTKEYWORD , "Supply Chains"))

The first step was to find relevant articles with information about SBOMs. The search words used were “sbom,” “software bill of material” and “software AND bill-of-material,” the last to broaden the search by allowing “software” and “bill-of-material” to appear separate in the text. The search strings were also limited to only include articles published after 2010, since that is the year the SPDX format was introduced, as well as to the subject area “computer science” and the language “english.” The last search also included the subject area “engineering,” once again to broaden the search. Both the second and fourth search string gave a larger number of hits, whereupon the keyword “Supply Chains” was added. Finally, the abstracts of the found articles were read through, and articles that did not match the field of interest were discarded. To find information specifically about SPDX the first search string was used. Most of these articles focused mostly on licenses but was used to gain insight into the SPDX community as well as methods to find SPDX files in code repositories.

To supplement the data found on Scopus a search on Google Scholar was conducted using the search string:

- "Software Bill of Material" AND "Opens Source software"

The search was also limited by Google Scholars filters to only include articles published between 2010 and 2023.

This resulted in a number of overlapping and new articles. As with the Scopus articles, the abstracts were read through and non interesting articles were discarded.

3.3 Archival study

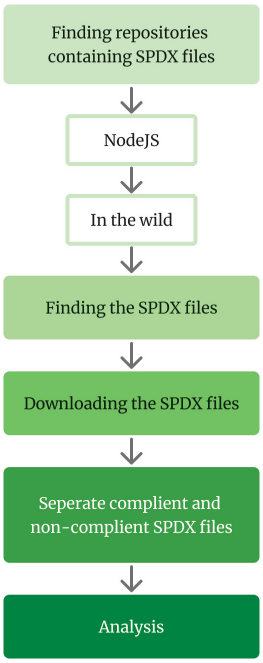

Figure 3.1: Archival study flowchart.

To answer the research questions an archival study was conducted, following the steps shown in Figure 3.1. An archival study entails analyzing documents and textual materials, including source code, textual data formats produced by and about organizations in various ways. Traditionally, archival studies apply to historical documents, but can also be used to study non-historical documents and texts, such as databases, emails, and web pages [20]. In this research, the documents that were collected and analyzed were SBOMs in the SPDX format. The first dataset was gathered from the top 1000 NodeJS OSS projects hosted

on GitHub and, due to an inadequate number of SBOMs found, the search was extended to include all public GitHub repositories on Sourcegraph. The SBOMs dataset was used to gather and summarize the information they contain, how they evolve by looking at versioning history, and how common the use of SBOM is.

3.3.1 Finding NodeJS repositories

The repositories were extracted on the 26th of March 2023, using the GitHub API[1] and the NPM registry API[2]. First, a search for the most starred repositories on GitHub was conducted using the query
`/search/repositories?q=stars:<fromStars>..<toStars>&type=repositories&per_page=100\&page=<page>`

Then, to verify that each repository actually was an NPM package, the full GitHub name from the GitHub API response was compared to the GitHub repository name found in the response from the query parameter `/<name>` against the NPM Registry API.

The GitHub API limits the number of returned repositories from a request to 1000. Therefore, the search had to be divided into several search intervals, by changing ”fromStars” and ”toStars” in the query, until 1000 NPM packages were found. The intervals used were decided based on results that returned less than a 1000 repositories. The number of actual found repositories is shown in the response, but since the API only grants access to the first 1000 of these, the number of returned repositories had to be less than 1000 to make sure no repositories were overlooked. Se A.1 for a list of the intervals used.

3.3.2 Finding repositories with SPDX files using Sourcegraph

To extend the search for SPDX SBOMs, a search for GitHub repositories with file content or path names containing SPDX files was conducted on the 11th of April 2023, using Sourcegraph [13].

In the SPDX specification v.2.3.0 a naming convention is suggested with the supported SPDX formats being *.spdx.yaml, *.spdx.yml, *.spdx.json, *.spdx.xml, *.spdx.rdf and *.spdx (tag:value). Searches for each extension were made on Sourcegraph. Note that the naming convention is just a suggestion, meaning there could be SBOMs in the SPDX format named in another fashion.

[1] https://docs.github.com/en/rest?apiVersion=2022-11-28
[2] https://github.com/npm/registry/blob/master/docs/REGISTRY-API.md

3.3.3 Finding SPDX files in the repositories

Two different searches were made for each NodeJS repository using the GitHub API. The first going through the repository itself looking for file paths containing the string "spdx", using the query
`/search/code?q=extension:spdx+extension:spdx.yml+extension:spdx.yaml+extension:spdx.json+extension:spdx.xml+extension:spdx.rdf+extension:spdx.json.sha256+repo:<owner/repository>`

The second search looked for files containing the word "spdx" or "sbom" amongst all the releases of the repositories using the query
`/repos/<owner/repository>/releases`
and then going through each release's assets. Going through the files found while searching for SPDX files using Sourcegraph, the names of the files found in the search were saved. A search for SPDX files in the releases of each repository was also done in the same fashion as for the NodeJs files. Finding the search words in the content of files in the repository could indicate that an SPDX file was produced at some point in the development process. We manually screened out examples or placeholders SBOM files.

3.3.4 Downloading the SPDX files

When the SPDX files and the repositories to which they belonged had been identified, they were downloaded using two approaches depending on whether they were placed in the repository itself or in the release (e.g., as tarball or other compressed archives). For the SPDX files placed in the repositories PyDriller [21] was used to go through all the commits of all the repositories and the source code of the file from each commit was saved. To download the SPDX files found in the releases, the download path for each version of each SPDX file was obtained from the Github API using the query `/repos/<owner/repository>/releases`, and going through each release containing the file.

3.3.5 Analysis

For the purpose of this research, the SPDX files found were divided into two groups, compliant and non-compliant SPDX files. The non-compliant SPDX files are files not containing any package information, since this study focuses on SBOMs for the purpose of keeping track of dependencies. The actual SBOMs include all files that at some point in time have contained information about at least one package. The number of packages described in each file was obtained using the open source tool SBOM Scorecard [22]. This is a command line tool that analyses the SBOM files according to five different subcategories, as shown in Table 3.1. It also returns

the total number of packages in SBOM files.

SBOM Scorecard was chosen since it provides a good insight into what information is included or not in the files. There is a very limited amount of tools of this kind, and to verify its usefulness we looked through the source code, and got an understanding of what the tool base the scores on. The tool is built to look at different file formats as well as different SPDX versions, which was important since we wanted to analyze all the SBOMs found, regardless of format and SPDX version, and still get an analogous result. When calculating the score for *Compliance* and *Creation Information* SBOM Scorecard looks at the fields in the SPDX Document Creation Information section. For the three package categories it looks at the Package Information section, assessing the different fields related to each category. It is not completely obvious why the various categories and field weight differently into the score, but the categories gives a good summery of the fields and looking at the total score we took care to look into which of the categories effected it.

RQ1 - Occurrence of SBOMs
To analyze the occurrence of SBOMs, all the found SPDX files were used and grouped according to their source, whether they were stored in a repository or part of a release, and whether they were actual SBOMs or not. The language of each repository containing a SBOM was also obtained from the response from the GitHub API using the query `/repos/<owner/repository>`.

RQ2 - Content of SBOMs
To examen the content of the SBOMs, the tool SBOM Scorecard was used. The SBOM Scorecard categories are each worth a certain amount of points which adds up to a total point for the SBOM file, as shown in Table 3.1. The results for each category is presented as a ratio of the max point for that category, where ratio 0 equals 0% and ratio 1 equals 100%. These results were analyzed for all compliant SPDX files. The difference in points between repository files and release files was assessed and values that diverged from the norm were noted.

Table 3.1: Max points for SBOM Scorecard categories.

Category	**Max Points**
Compliance	25
Package Identification	20
Package Version	20
Package Licences	20
Creation Info	15
Total	**100**

RQ3 - How the SBOMs evolve
To examine how the SBOMs have evolved over time, SBOM Scorecard was used on all versions of all SBOMs and then the changes in *Total Ratio* over time were analyzed. Going through all the latest versions of the found SBOMs the SPDX version and creation date were noted to determine which SPDX version was used the most and whether the SBOM files used the latest SPDX version available at the time of creation. This to gain an understanding of the popularity of each SPDX version as well as how well the SBOMs are kept up to date with the latest SPDX version. The creation information in each SBOM was also noted and compared between the different languages of the repositories where the SBOMs were found.

3.4 Validity and reliability

There was a risk that the data collected when searching for SBOMs in the top 1000 NodeJS repositories were not sufficient for answering RQ2 and RQ3. This was remedied by making an additional search for SBOMs in other ecosystems as well.

A big challenge was to extract the data from the NPM Registry API and the GitHub API. The APIs have limitations in the amount of data returned from queries, making for a time-consuming mining process. We did find ways to get around the limitation, for example, we combined the use of the GitHub API and the NPM Registry API in our efforts to find the most starred NPM packages. When extending the search for SBOMs it was clear that GitHub was no longer an option, since it only returns the top 1000 results, and as done with the stars search there was no possible way of limiting intervals here. We instead used Sourcegraph which, while somewhat limiting the number of available repositories, made it possible to quickly find all files included in the search strings.

Another obstacle we had to consider was that a lot of the SPDX files were non-compliant, with most of them only used to convey information about the project's license. This made them valid SPDX files, but not valid SBOMs. To make sure that our statistics about SBOMs were correct, we made sure to identify these and exclude them when answering RQ2 and RQ3.

Before mining software repositories, we had to consider where to find the SBOMs. As Stoddard et al. [23] point out there are multiple different ways for software providers to share their SBOMs. Some use email, others place the SBOM inside the software source code or on their website, or the provider can create a whole service dedicated to sharing the SBOMs and keeping the users aware of new updates. In other words, there is no standard way of sharing SBOMs. However, since we used GitHub as our main source, we chose to look for SBOMs in the two places they could be found in a GitHub repository, in the source code, and in the releases.

Chapter 4

Results and Analysis

4.1 RQ1 - How common is the occurrence of SBOMs in OSS projects hosted on GitHub?

4.1.1 Visualisation of the data

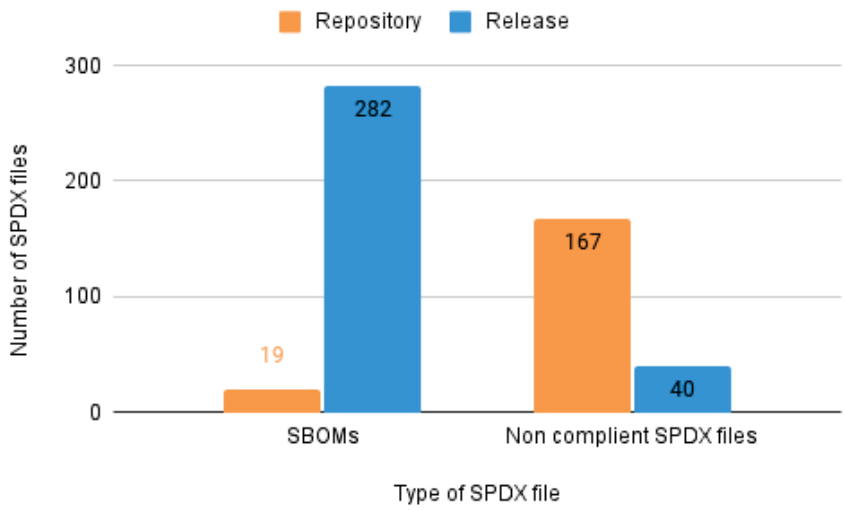

Figure 4.1: The total number of SPDX files found, their source, and number of SBOMs in comparison to non-compliant SPDX files.

The mining of SBOMs from Sourcegraph and NPM resulted in a total of 508 SPDX files, of which 506 were found in the Sourcegraph search, one was found in the NPM search and one was found in both searches. The graph in Figure 4.1 shows the number of actual SBOM files found compared to the number of non-compliant SPDX files and how many of the respective file types was found in the repository itself versus files found in the project release. Of the total number of SPDX files,

301 were actual SBOMs, and of these 19 were found in repositories and 282 in releases.

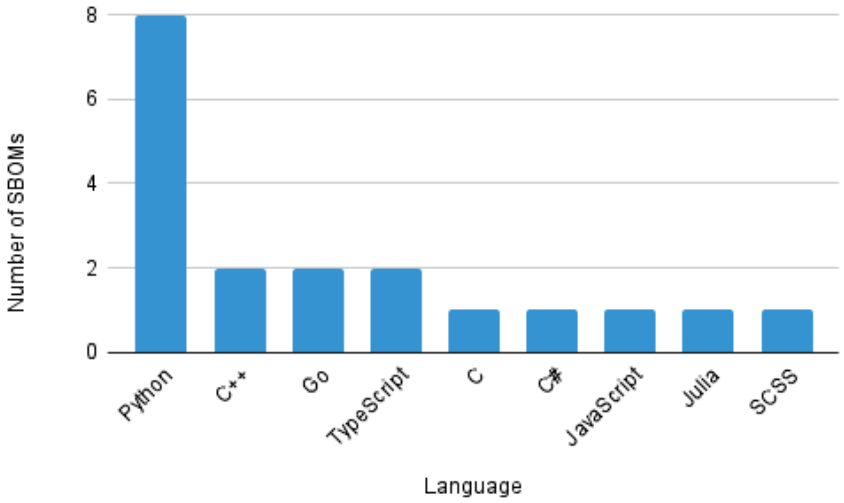

Figure 4.2: Number of repository SBOMs found per language.

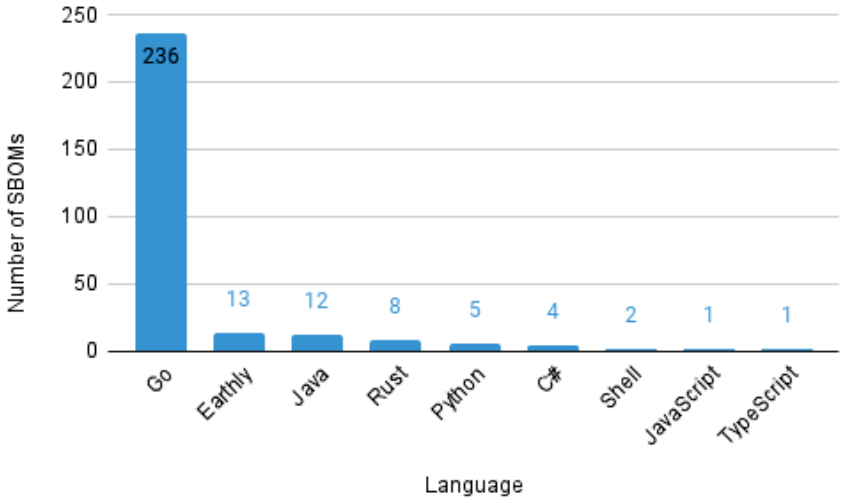

Figure 4.3: Number of release SBOMs found per language.

The distribution of found SBOMs, excluding non-compliant SPDX files, amongst different programming languages is displayed in Figures 4.2 and 4.3. Amongst the repositories where the SBOMs were found in the repository, Pyhton was the most

common language followed by C++, Go, and TypeScript. For SBOMs found in the release, Go accounts for 236 out of a total of 282 releases.

4.1.2 Answer to the research question

There were two different approaches used to find SBOMs for the study. The first was looking through the 1000 most popular NPM packages based on GitHub stars. Amongst these, only 0.2% of the packages had a SBOM. The NPM registry has over two million packages [1] and given that so few SBOMs were found among the most popular packages, one can assume that an even smaller percentage of the total packages on NPM contains one. The second approach for obtaining SBOMs was using Sourcegraph, which contains about 1.8 million GitHub repositories. However, it can be the case that not all the packages associated to these repositories need a SBOM. Amongst all these repositories, 507 SPDX files were found, of which 59% were compliant SPDX files containing information about packages, while 41% were non-compliant SPDX files.

Looking at the distribution of SBOMs among the different programming languages, a clear majority of the repositories where the SBOM were found in the release were written in Go. Amongst the repositories where the SBOM were placed in the repository itself the distribution of languages where a bit more even, although Python stands out.

To conclude, SBOMs are not commonly shared in OSS projects. When shared, almost half do not contain valuable information. Projects developed using Go or Python tends to share SBOMs more than projects in other programming languages.

4.2 RQ2 - To what extent do SBOMs in projects hosted on GitHub follow the SPDX standard?

4.2.1 Visualisation of the data

To answer RQ2, we will exclude non-compliant SPDX files from the result. Therefore, SBOMs mentioned in this section are exclusively compliant SPDX files.

Below the SBOM Scorecard results will be showcased by category, comparing the results for SBOM files collected from repositories vs. SBOM files collected from releases. As stated in section 3.3.5, the total score from SBOM Scorecard consists of five subcategories. The *Compliance* category assesses the very minimum requirements for an SPDX file. For the purpose of this research paper, these requirements have been used when collecting SBOM files. Therefore, all files in

[1] https://www.npmjs.com/

the dataset get full points for this category and no further results are shown for *Compliance*.

Package Version Ratio

Figure 4.4: Package Version Ratio for each SBOM found in repositories.

Figure 4.5: Package Version Ratio for each SBOM found in releases.

For the category *Package Version*, the majority of both repository SBOM files and release SBOM files gets a point ratio of 1. Repository SBOMs have an average point ratio of 0.65 whereas release SBOMs have an almost perfect average point ratio of 0.97.

Package Identification Ratio

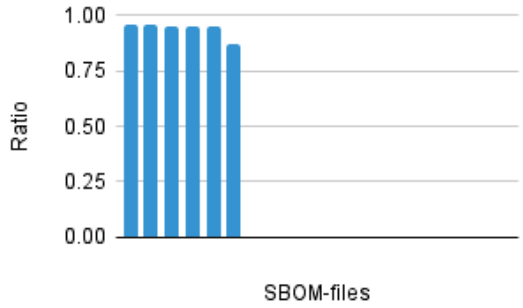

Figure 4.6: Package Identification Ratio for each SBOM found in repositories.

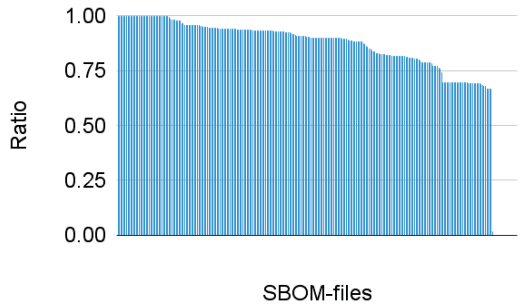

Figure 4.7: Package Identification Ratio for each SBOM found in releases.

When it comes to *Package Identification Ratio*, the graphs show a clear difference between repository SBOMs and release SBOMs. The majority of release SBOMs

have a point ratio above 0.75 with an average of 0.82, while the majority of repository SBOMs have a point ratio of 0, with the average point ratio being 0.30.

Package Licence Ratio

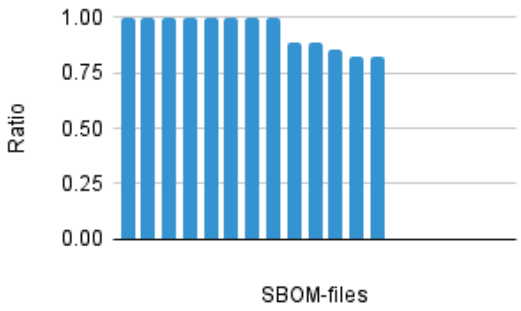

Figure 4.8: Package Licence Ratio for each SBOM found in repositories.

Figure 4.9: Package Licence Ratio for each SBOM found in releases.

Package Licence is the only category where repository SBOMs score higher than release SBOMs. Repository SBOMs have an average point ratio of 0.65 while release SBOMs have a shockingly low average of 0.13.

Creation Info Ratio

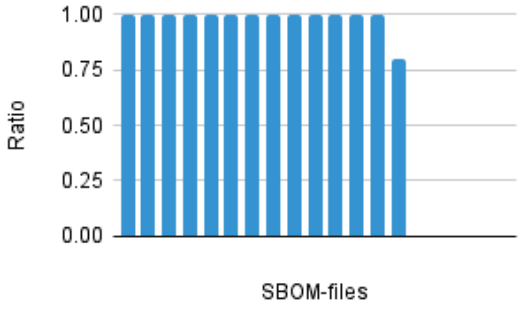

Figure 4.10: Creation Info Ratio for each SBOM found in repositories.

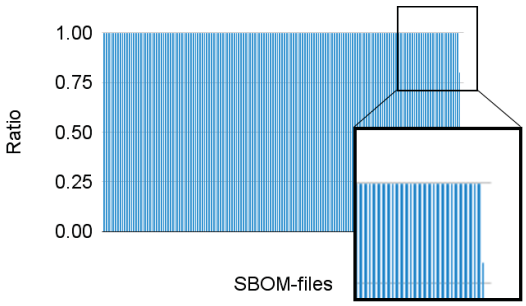

Figure 4.11: Creation Info Ratio for each SBOM found in releases.

The results for *Creation Info* are high for both repository SBOMs and release SBOMs. Repository SBOMs have an average point ratio of 0.73 and release SBOMs have an average of 0.99.

Total Ratio

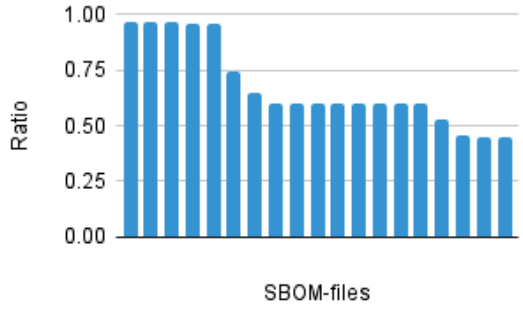

Figure 4.12: Total Ratio for each SBOM found in repositories.

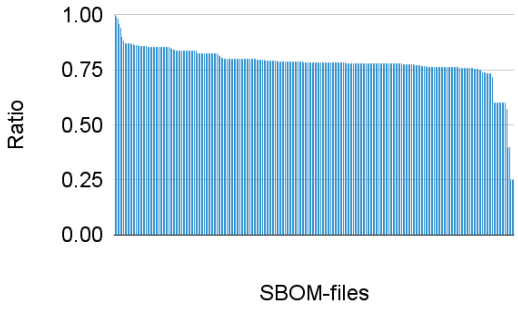

Figure 4.13: Total Ratio for each SBOM found in releases.

As shown by the trend for the subcategories the *Total Ratio* clearly has a higher score for release SBOMs, with the majority having a point ratio above 0.75 and the average point ratio being 0.78. Looking at the *Total Ratio* for repository files the majority are above 0.5 and the average point ratio is 0.68.

4.2.2 Answer to the research question

The SPDX standard is a format for what information a SBOM file should and can include. As stated in Chapter 1, there is one mandatory section, with mandatory and optional fields, and seven optional sections. For the purpose of answering this research question, we have used the tool SBOM Scorecard which reports information regarding the categories *Specification Compliance*, *Creation Info*, and *Package Info*. These individual category scores are combined into a total score for the SBOM file. *Package Info* is divided into three sub-categories which will only give points if the file contains information about packages. For answering this question, we have only considered SBOMs that contain packages, so they will at the minimum have points for one of these subcategories.

Table 4.1: Compilation of SBOM Scorecard results per category presented in ratio.

Category	**Repository**			**Release**		
	Min	**Max**	**Average**	**Min**	**Max**	**Average**
Compliance Ratio	1	1	1	1	1	1
Package Identification Ratio	0	0.9552	0.2965	0	1	0.8248
Package Version Ratio	0	1	0.6542	0	1	0.9723
Package Licenses Ratio	0	1	0.6456	0	1	0.1286
Creation Info Ratio	0	1	0.7263	0	1	0.9887
Total Ratio	0.45	0.9672	0.6782	0.25	0.9986	0.7835

As shown in Table 4.1, all of the SBOMs in the data set have maximum points for *Compliance* which means that they are valid SPDX files. For this category, it is only possible to receive etiher 0 or 1 points for ratio. Since one of the requirements when collecting the dataset was that the SBOM files are valid SPDX files, it is expected that all files receive the max point here. When considering the other categories, a more varied average between repository SBOMs and release SBOMs can be noted, with release SBOMs having a higher average point ratio for all categories but *Package Licence*. None of the repository SBOM received a point ratio of 1 on *Package Identification*, instead the highest point ratio is 0.9552. On all other fields, there is at least one SBOM file that received the max point.

As stated above the only possible point ratio for *Compliance* is either 0 or 1. The other categories have a more varied point ratio due to how the ratio is calculated. When it comes to the three package categories, they are created by counting the combined point ratio for all packages and then splitting this combined ratio by the total amount of packages in the SBOM file. The *Creation Info Ratio* is created by giving the point ratio 0.6 if the SBOM file has a creator or a creator tool, then 0.2 points are added respectively for creation date and creation tool version.

To summarize, the release SBOMs follow the SPDX standard to a higher degree than the repository SBOMs. Looking at the subcategories, excluding *Compliance*, *Package Licenses* has the overall lowest average while *Creation Info* has the overall highest average.

4.3 RQ3 - How do SBOMs evolve in OSS projects hosted on GitHub?

4.3.1 Visualisation of the data

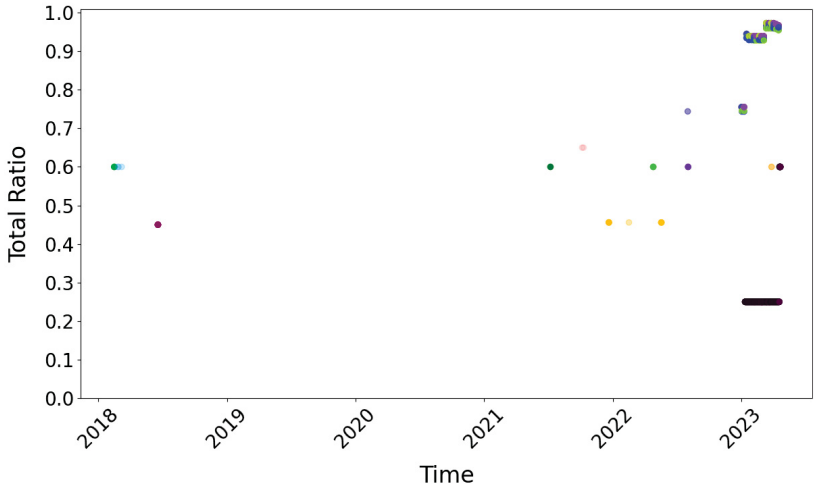

Figure 4.14: Change in SBOM Scorecard *Total Ratio* over time for the repository SBOMs. Each color represent one SBOM at different points in time.

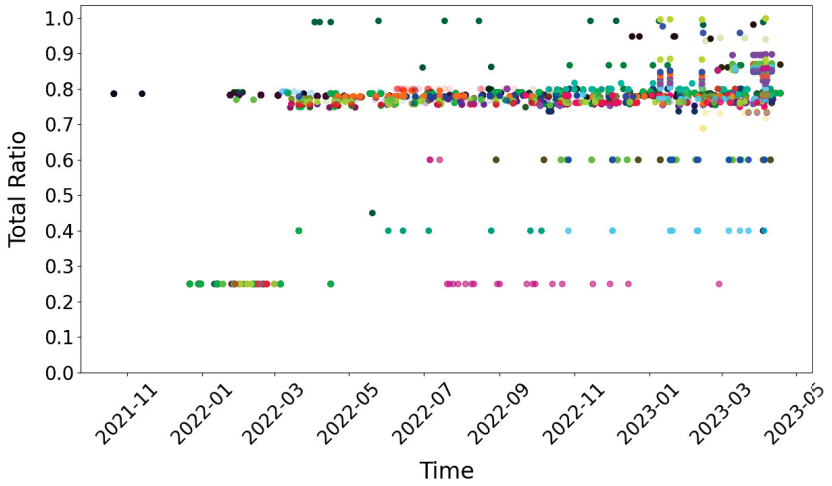

Figure 4.15: Change in SBOM Scorecard *Total Ratio* over time for the release SBOMs. Each color represent one SBOM at different points in time.

The graphs in Figure 4.14 and 4.15 shows how the *Total Ratio* has changed over time. As mentioned in Section 4.2.1 all files examined have the *Compliance Ratio* 1 and since this makes up 25% of the total score, all the SBOM files have a *Total Ratio* of at least 0.25. This means that the files with exactly 0.25 points score zero points in all other categories, meaning that they do not include information about any packages.

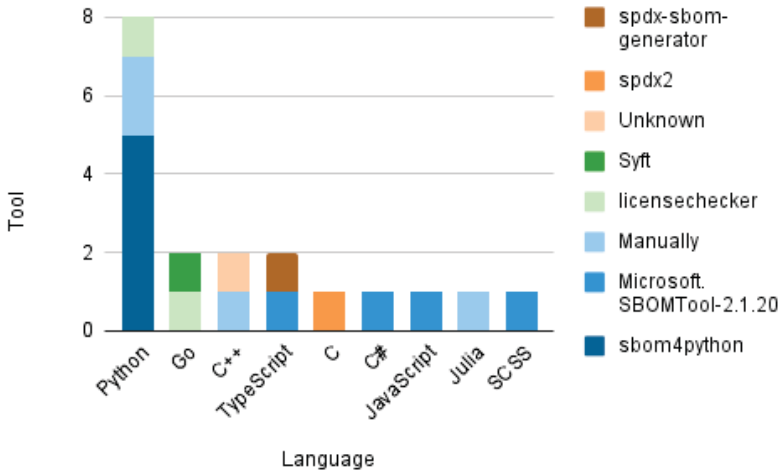

Figure 4.16: Distribution of tools used for creating the repository SBOMs per language.

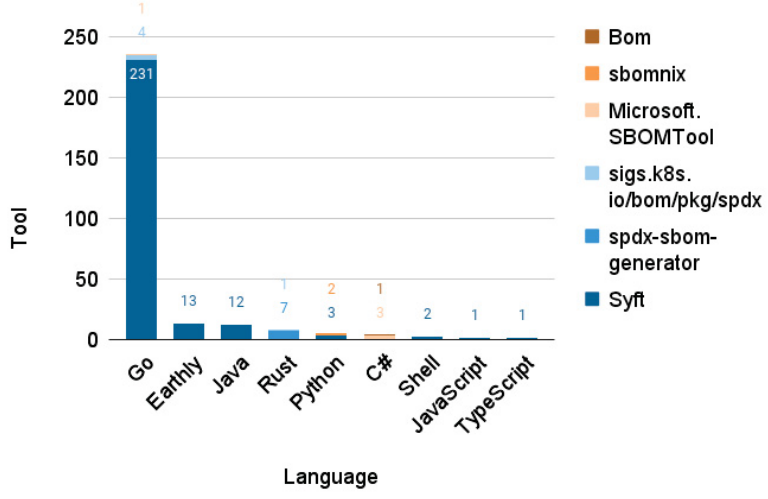

Figure 4.17: Distribution of tools used for creating the release SBOMs per language.

The graphs in Figure 4.16 and 4.17 shows what tools are used when creating SBOMs and how they are distributed between the different programming languages. The most popular tool amongst the release SBOMs is Syft with 93% using it. Amongst the repository SBOMs, however, Syft is only used to create one of the SBOMs, and instead sbom4python is the most common one with 26% using it.

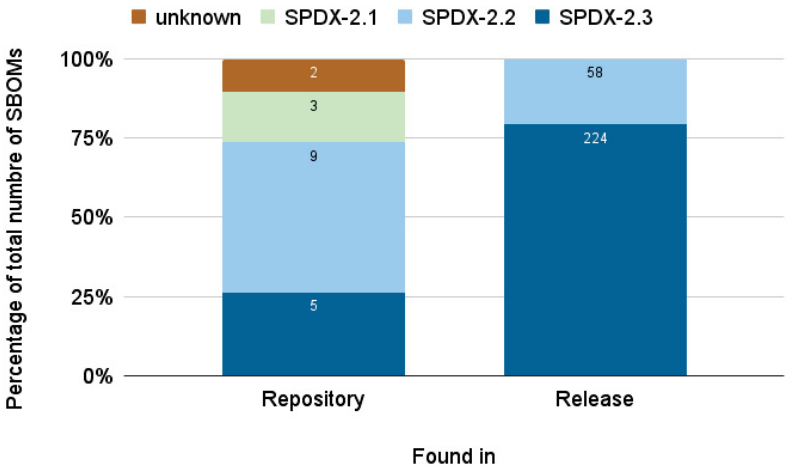

Figure 4.18: Distribution of the SPDX versions of the repository SBOMs compared to the release SBOMs.

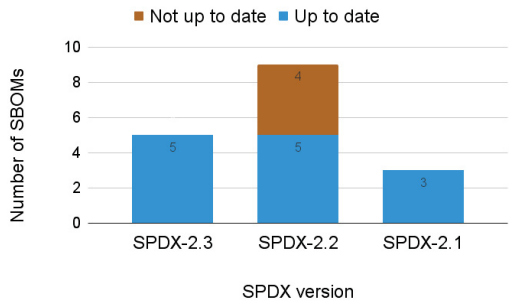

Figure 4.19: Number of repository SBOMs with an SPDX version that is up to date.

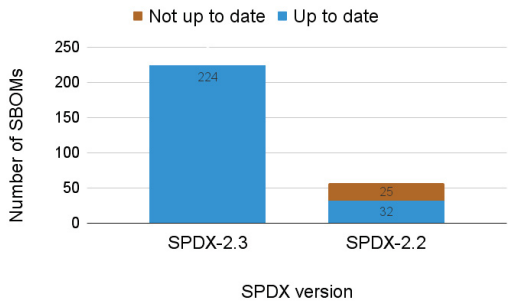

Figure 4.20: Number of release SBOMs with an SPDX version that is up to date.

The graph in Figure 4.18 shows the distribution of different SPDX versions used in the repository SBOMs compared to the release SBOMs. Figure 4.19 and Figure 4.20 further shows whether the SPDX version used is the latest version available at the time the SBOM was created.

4.3.2 Answer to the research question

As was shown in Figures 4.16 and 4.17, the tools used for generating SBOMs in the repositories compared to releases differ. Syft is the by far the most common among the release SBOMs, while sbom4python is the most common amongst the repository SBOMs, which also happens to be a language-specific tool for the most used language in our dataset.

Comparing tools used for creating repository SBOMs to those used for creating release SBOMs, only MicrosoftSBOMTool, Syft, and spdx-sbom-generator are used for both. It is also worth mentioning that none of the SBOMs found in the releases were created manually or did not specify how they were created, while 26% of the repositories SBOMs were either created manually or did not provide any creation information.

Taking a closer look at how the SBOMs evolved over time, we can see that 99.4% of the SBOMs analyzed in this research had their first version introduced after May of 2021. This is interesting since this was when the Biden Administration's executive order was introduced.

It is evident that most of the repository SBOMs have not been updated more than a couple of times. Five out of the 19 SBOMs are only included in one commit, and eight are included in five commits or less. However, it is important to note that only one of these eight repositories has been updated since the latest commit including the SBOM. Therefor we can not draw a conclusion whether the SBOMs are updated continuously as the project is developed or not. Another interesting observation is that the *Dynamics365Commerce* repositories, all owned by Microsoft, had a *Total Ratio* of 0.25 for 101 out of 104 commits, before the *Total Ratio* suddenly increased, meaning package information was added at this point in time.

There are also five SBOMs that have a much higher *Total Ratio* than the rest of the dataset, all belonging to the repository `intel/cve-bin-tool`. These follow the same curve, though they differ slightly in *Total Ratio*.

SBOM files found in releases are updated much more frequently and have a *Total Ratio* which averages at 0.73 compared to the average for repository files which is 0.41. A small part of the SBOMs has a lower *Total Ratio* of 0.4 or 0.6. Taking a closer look at the SBOM Scorecard categories, using the data in Appendix A.2, all the SBOMs with a *Total Ratio* of 0.4 have a full score on *Compliance* and *Creation Info*. The SBOMs with a *Total Ratio* of 0.6 have an additional full score on either *Package Version* or *Package Licenses*. There are two files standing out with their high ratio. `Keptn-sbom-bridge2.spdx.json` with its average *Total Ratio* at 0.99 and `docker-openvpn-client/jsloan117-docker-openvpn-client.spdx.json`, which started out with a *Total Ratio* of 0.25 but on the second commit improved to 0.99 and has since kept that average. There are a few more files that have improved in a similar manner, starting out at 0.25 and then increasing within one

or a couple of releases. However, none of them have reached the same high *Total Ratio*. A majority of the SBOMs stay consistent in their scores after reaching a certain point, with a few exceptions. Nine SBOMs, all belonging to the repository `goreleaser/goreleaser`, start out at a *Total Ratio* of 0.25 then increase to 0.78 before it dips down to 0.4 and then yet again increases to 0.78. These changes in *Total Ratio* are due to *Package Information* and *Creation Information* being added, then the packages are removed before being added once again.

Four SBOMs that, in contrast to the rest, get a worse score over time are `constellation.spdx.sbom` which starts out at an average *Total Ratio* of 0.78 and then plunges to 0.4, and three SBOMs from the repository `microsoft/sbom-tool` which all start out on an average at 0.6 and then decrease to 0.25. This decline in *Total Ratio* is caused by the removal of *Package Information* in all cases.

Table 4.2: SPDX versions release dates. [24]

Version	**Release date**
SPDX-2.3	2022-11-03
SPDX-2.2	2020-06-04
SPDX-2.1	2017-07-03

Considering the use of different SPDX versions, the SPDX versions used by release SBOMs were SPDX-2.2 and SPDX-2.3. This is to be expected since the oldest release SBOM was created on 2022-05-26, which is approximately two years after SPDX-2.2 was released. SPDX-2.1, SPDX-2.2, and SPDX-2.3 were found amongst the repository SBOMs where the oldest was created on 2018-02-15, about two years before SPDX-2.2 was released.

90% of the SBOMs SPDX versions were up to date, and amongst the ones that were not, all followed SPDX-2.2. Of the SPDX-2.2 SBOMs, 56% were up to date.

In conclusion, the majority of the SBOMs found in this study are updated frequently and improve or remain consistent over time. The most common SPDX version amongst the SBOMs is SPDX-2.3 and the SPDX version used is generally the latest SPDX version available at the time of creation. Amongst the tools used to create SBOMs, Syft is by far the most popular.

Chapter 5

Discussion

This study aims to give insights into the occurrence of SBOMs in OSS projects, what they contain, and how they evolve. It looks specifically at NPM packages and then widens the scope to look for SBOMs in other ecosystems as well. The total number of actual SBOMs found was fewer than we had anticipated. According to the Linux Foundation research on SBOM readiness in January of 2022, 60% of the participating organization consider that the use of SBOMs in OSS is equally as important as in proprietary software, and 29% believe that SBOMs are more important for OSS. Based on its research, the Linux Foundation estimated that 78% of organizations would use SBOMs in 2022 [5]. Therefore, one could assume that the occurrence of SBOMs in OSS projects would be somewhat higher than this research shows. It is however important to note that the Linux Foundation does not distinguish between different formats, while this study only looks at the SPDX format. Further, in contradiction to the Linux Foundation survey, both Xia et al. [14] and Zahan et al. [10] do not provide as positive of an image of the current state of SBOM usage. Xia et al. [14] specifically point out that the usage of SBOMs in OSS projects is even lower than the overall user.

The majority of the release SBOMs were found in Go-projects, while Python was the language that had the most repository SBOMs. Why the use of SBOMs is more common in these languages could have different explanations. It might be that the tools are better suited for these languages. Over half of the Python repositories used a language-specific tool. Another possible explanation could be that these communities might be more security oriented and therefore are ahead of the curve when it comes to SBOM usage, or perhaps more security focused programs are created using Go or Python.

Of all SPDX files found, 40.7% were non-compliant and out of them, 80.7% were found in the repository. The majority of the SBOMs found were placed amongst the assets in the releases. This could be explained by the convenience of the SBOM being automatically updated for each new release. Having each version of the SBOM corresponding to a release could in turn benefit version control management. One could assume this would make it easier to identify when a potentially harmful

package was added.

Looking at the SPDX version, the majority were up to date, and it seems most have left SPDX2.1 behind. The 29 SBOMs that were not up to date all followed the SPDX 2.2 specification. Why this is the case, would be interesting to look into further. It could simply be that the creators have forgotten to update or are not aware that there is a later version. However, it could also be a conscious decision. The SPDX standard is constantly evolving to include the information that the SPDX community find helpful [19]. That 44% have chosen not to update could therefore indicate that they do not find SPDX-2.3 to be an improvement. Regardless of the cause, the OSS community using different SPDX version could lead to communication issues between SBOM providers and consumers. They may deem different information to be valuable, and information the consumers may need could be missing.

Looking at the content of the SBOMs using SBOM Scorecard, there are some noticeable differences between the SBOMs found in the repositories and the ones found in the releases. Release SBOMs often have a higher average score for all categories except for *Package License* were it has a considerably lower average. This result has an impact on projects and projects' maintainers since not following the third-party license obligations or using a package without a license could lead to severe legal actions [25]. We hypothesize that the low license score is caused by the tool used, whether the SBOM is created manually or not, the creators' preferences, or the packages not having a license. The *Total Ratio* for the SBOMs also mostly stays consistent over time. This could be that the information included is deemed to be adequate. It could also mean that the maintainers have chosen a tool and then not given more thought into what fields are actually included or not.

5.1 Conclusion

This study provides a quantitative insight into the current state of SBOM occurrence in OSS projects as well as how they evolve over time. It was found that a very small fraction of OSS projects actually provide a SBOM, and they can be found both in the source code of the repositories and amongst the assets of the releases, though most were found in the releases. An interesting note is that 99.4 % of the SBOMs first appeared after the Biden Administration's executive order was introduced, validating the effect it has had on the software community.

Regarding the SPDX standard, the SBOMs found in the releases follow it to a higher degree than those found in the repositories, and most use the latest version, SPDX-2.3.

The majority of the SBOMs were updated frequently and the most popular tool for creation is Syft.

5.2 Future work

The usage of SBOMs is such an unexplored subject, accordingly; there are several outlets for further investigation. First, assessing the occurrence of SBOMs in OSS projects in upcoming years, to see how and if the usage of SBOMs has increased and how future forecast for SBOM usage look. One could also extend the scope to include SBOMs of other formats and compare their popularity and growth.

Further, one could look closer at the content of the SBOMs and give an answer to why certain fields are more often excluded or included. Does this vary between different formats and tools used for generating the SBOMs?

The choice of SPDX version could be further investigated by interviewing maintainers, to gain a qualitative understanding of the motivation for updating the versions or not.

We also suggest that future research looks into the Go and Python development communities to empirically understand what factors motivate their extensive use of SBOM with respect to other communities.

5.3 Limitations

There are a few limitations to the search for SBOMs. We based the NPM search on the most starred repositories on GitHub. These packages might be quite old and no longer active, since it takes time to gain that many stars, thus they are less likely to have incorporated SBOMs. However, it can be assumed that most of the popular packages are still active and would therefore still be representative.

The other limitation regarding the search for SBOMs is that in the extended search, Sourcegraph was used to search for GitHub repositories. Sourcegraph does not provide access to all public repositories, therefore it is possible that we do not cover all OSS projects that use GitHub as a code-sharing platform. We did, however, search through more than 1.8 million repositories, and the result from the NPM search gives a good indication that SBOMs are not commonplace, at least not in the NodeJS ecosystem.

Further, this study looks into whether the SBOMs are present in repositories or releases and if they are continuously updated. It does not, however, verify whether the packages stated in the SBOMs are corresponding to the packages included in the project. Neither does the study certify whether the information needed for the SBOM is actually available in the different packages. To what extent the SBOMs are a correct representation of the incorporated packages would provide more insight into how well the SBOMs are maintained.

References

[1] J. Thomas and J. New. (2022) Open source: Understanding the software supply chain is key to improving security. [Online]. Available: https://www.ibm.com/policy/open-source-security/ [Accessed: 2023-04-09].

[2] Microsoft. (2022) Supply chain attacks. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/security/intelligence/supply-chain-malware?view=o365-worldwide [Accessed: 2023-04-09].

[3] S. Oladimeji and S. M. Kerner. (2022) Solarwinds hack explained: Everything you need to know. [Online]. Available: https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know [Accessed: 2023-04-09].

[4] M. Arvidsson, "Vad du behöver veta om ransomware-attacken mot kaseya"," *inuit.se*, 2022. [Online]. Available: https://www.inuit.se/blogg/vad-du-behover-veta-om-ransomware-attacken-mot-kasey [Accessed: 2023-04-09].

[5] S. Hendrick, VP Research and The Linux Foundation, "Software bill of materials (sbom) and cybersecurity readiness," *The Linux Foundation*, 2022. [Online]. Available: https://www.linuxfoundation.org/research/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness [Accessed: 2023-01-29].

[6] L. S. Vailshery, "Most used web frameworks among developers worldwide, as of 2022," *statista.com*, 2022. [Online]. Available: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ [Accessed: 2023-04-09].

[7] Stack Overflow. (2022) 2022 developer survey. [Online]. Available: https://survey.stackoverflow.co/2022/#technology-version-control [Accessed: 2023-05-14].

[8] B. Bensing, "History of software bill of material (sbom)," *itrevo-*

lution.com, 2022. [Online]. Available: https://itrevolution.com/articles/history-of-software-bill-of-material-sbom/ [Accessed: 2023-04-09].

[9] Linux Foundation and its Contributors, "Spdx specification v2.3.0," *spdx.github.io*, 2022. [Online]. Available: https://spdx.github.io/spdx-spec/v2.3/ [Accessed: 2023-04-09].

[10] N. Zahan, E. Lin, M. Tamanna, W. Enck, and L. Williams, "Software bills of materials are required. are we there yet?" *IEEE Security Privacy*, vol. 21, no. 2, pp. 82–88, 2023.

[11] J.Potter, "npm vs pnpm vs yarn," *npmtrends.com*, 2023. [Online]. Available: https://npmtrends.com/npm-vs-pnpm-vs-yarn [Accessed: 2023-04-09].

[12] A. Muminovic, "npm vs pnpm vs yarn," *medium.com*, 2019. [Online]. Available: https://medium.com/maestral-solutions/alternative-package-managers-for-node-js-f52805b98064 [Accessed: 2023-04-09].

[13] Sourcegraph. Sourcegraph. [Online]. Available: https://sourcegraph.com/search? [Accessed: 2023-05-11].

[14] B. Xia, T. Bi, Z. Xing, Q. Lu, and L. Zhu, "An empirical study on software bill of materials: Where we stand and the road ahead," 2023.

[15] S. Carmody, A. Coravos, G. Fahs, A. Hatch, J. Medina, B. Woods, and J. Corman, "Building resilient medical technology supply chains with a software bill of materials," *npj Digital Medicine*, vol. 4, no. 34, pp. 2398–6352, 2021.

[16] S. Nadgowda and L. Luan, "Tapiserí: Blueprint to modernize devsecops for real world," in *Proceedings of the Seventh International Workshop on Container Technologies and Container Clouds*, ser. WoC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 13–18. [Online]. Available: https://doi-org.miman.bib.bth.se/10.1145/3493649.3493655

[17] S. Nadgowda, "Engram: The one security platform for modern software supply chain risks," in *Proceedings of the Eighth International Workshop on Container Technologies and Container Clouds*, ser. WoC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 7–12.

[18] R. A. Martin, "Visibility control: Addressing supply chain challenges to trustworthy software-enabled things," in *2020 IEEE Systems Security Symposium (SSS)*, 2020, pp. 1–4.

[19] R. Gandhi, M. Germonprez, and G. J. P. Link, "Open data standards for open source software risk management routines: An examination of spdx," in *Proceedings of the 2018 ACM International Conference*

on Supporting Group Work, ser. GROUP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 219–229. [Online]. Available: https://doi-org.miman.bib.bth.se/10.1145/3148330.3148333

[20] J. Mohr and M. Ventresca, "Archival research methods"," *Blackwell companion to organizations*, pp. 805–828, 2022.

[21] D. Spadini, M. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining soft-ware repositories," 2018. [Online]. Available: https://github.com/spdx/spdx-spec/releases

[22] eBay. Sbom scorecard. [Online]. Available: https://github.com/eBay/sbom-scorecard [Accessed: 2023-05-11].

[23] J. T. Stoddard, M. A. Cutshaw, T. Williams, A. Friedman, and J. Murphy, "Software bill of materials (sbom) sharing lifecycle report," 4 2023. [Online]. Available: https://www.osti.gov/biblio/1969133

[24] SPDX. spdx-spec. [Online]. Available: https://github.com/spdx/spdx-spec/releases [Accessed: 2023-05-11].

[25] P. Odence, "Five types of software licenses you need to understand," *synopsys.com*, 2022. [Online]. Available: https://www.synopsys.com/blogs/software-security/5-types-of-software-licenses-you-need-to-understand/ [Accessed: 2023-05-09].

Appendix A

A.1 NPM Regestry search intervals

The intervals used for the NPM Reqistry search were as follows:

- 400 000 - 20 000
- 19 999 - 13 000
- 12 999 - 10 000
- 9999 - 8100
- 8099 - 7000
- 6999 - 6100
- 6099 - 5400
- 5399 - 4800
- 4799 - 4400
- 4399 - 4000

A.2 All data collected about each SBOM

All data collected about the SBOMs using the GitHub API, SBOM Scorecard and ad-hoc scripts can be found in this external appendix:
https://github.com/VeronicaAxelsson/SBOMs-in-OSS

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden