Wild SBOMs: a Large-scale Dataset of Software Bills of Materials from Public Code

Luís Soeiro*, Thomas Robert*, Stefano Zacchiroli*
*LTCI, Télécom Paris, Institut Polytechnique de Paris, France
{luis.soeiro, thomas.robert, stefano.zacchiroli}@telecom-paris.fr

***Abstract*—Developers gain productivity by reusing readily available Free and Open Source Software (FOSS) components. Such practices also bring some difficulties, such as managing licensing, components and related security. One approach to handle those difficulties is to use Software Bill of Materials (SBOMs). While there have been studies on the readiness of practitioners to embrace SBOMs and on the SBOM tools ecosystem, a large scale study on SBOM practices based on SBOM files produced in the wild is still lacking. A starting point for such a study is a large dataset of SBOM files found in the wild. We introduce such a dataset, consisting of over 78 thousand unique SBOM files, deduplicated from those found in over 94 million repositories. We include metadata that contains the standard and format used, quality score generated by the tool *sbomqs*, number of revisions, filenames and provenance information. Finally, we give suggestions and examples of research that could bring new insights on assessing and improving SBOM real practices.**

***Index Terms*—SBOM dataset, SBOM standards, SBOM usage in the wild, SBOM scores**

I. Introduction

Modern software development reuses relevant amounts of code from third parties, mainly in the form of Free and Open Source (FOSS) code, to build applications [1]. FOSS ecosystems can be very large, from thousands of components to tens of thousands when accounting for their dependencies [2]. While gains from reusing FOSS code can reach up to trillions of dollars [3], there are also some difficulties to be handled, such as licensing requirements [4], repository and component governance [5], and security considerations [6]. One of the mechanisms that have been proposed to aid with those issues and to provide more transparency to software projects is the use of Software Bill of Materials (SBOM) [7], an inventory of all third-party components and dependencies used in an application. [8].

Since the publication of the Software Identification Tags (SWID Tags) in 2009 [9], the Software Package Data Exchange (SPDX) SBOM introduction in 2010 [10], and the CycloneDX SBOM standard prototype in 2017 [11], SBOMs have gained traction. In the wake of major software supply chain security incidents (e.g., SolarWinds, Log4J), the US National Telecommunications and Information Administration (NTIA), following the U.S. Presidential Order 14028, started to promote the use of SBOMs [7], and it has become a requirement for supplying software to the U.S. government [12]. In the E.U., the Cyber Resilience Act (CRA) proposal of 2022 also brings a similar requirement [13].

There has been studies on the benefits and challenges of SBOM adoption [12], [14], on the practitioners' views on the subject [7], and on the difficulties of generating correct SBOM files using SBOM creation tools [15]. According to a Linux Foundation's survey from 2022, 20% of the organizations interviewed are already producing SBOMs and 40% are consuming them in production [16]. Research on real SBOM files found in the wild can help to evaluate the state of those practices. Existing SBOM datasets mined from repositories contain a maximum of 1,151 files, but lack diversity (see VI). Easily accessible, more diverse datasets, with more samples are needed [17]. Analysis of such a corpus can provide insights to real world SBOM usage aspects, including standards adherence and overall quality. Additionally, it can facilitate the SBOM tools ecosystem, by providing developers with material to test, evaluate and benchmark tools.

Contributions and use cases: We introduce a large and diverse dataset of SBOM files that came from public version control systems (VCS). The package is comprised of two parts:

1) A deduplicated dataset of 78,612 unique SBOM files, that were found on 94,618,356 unique source repositories, distributed in 1,782 unique forges and package repositories (*forges* for short);
2) Mined metadata, including the SBOM standard adopted (e.g., SPDX, CycloneDX), the file format (e.g., xml, json, tag-value, yaml), a quality score generated by the *sbomqs* tool [18], provenance information for each SBOM file, and all the different filenames each one had being observed with.

The dataset can be used to support use cases like: (a) large scale analysis of SBOM adoption, most used standards, and file formats, possibly segmented by originating forge, creator tool and other properties; (b) analysis of SBOM quality in the wild; (c) benchmarking of the SBOM tools, by evaluating their effectiveness and correctness of their functionalities, such as SBOM consumption, conversion, and validation; (d) software composition analysis; (e) vulnerability analysis.

Data availability: The dataset is released as open data, and the related code as free and open source code. The

Supported by the industrial chair Cybersecurity for Critical Networked Infrastructures (cyberCNI.fr) with support of the FEDER development fund of the Brittany region, France. This work was made possible by Software Heritage, the great library of source code: https://www.softwareheritage.org. Special thanks to Valentin Lorentz for his help with data extraction.

replication package allows the dataset to be recreated from scratch. It is available from Zenodo [19], as a tar archive containing a directory tree with all the SBOM files, a set of CSV files containing the metadata, the software used to generated it, and detailed usage and reproducibility instructions (*see README.md* file).

II. Methods and Reproducibility

We have adopted the following criteria for building the dataset: (a) diversity (i.e., it should include as many different forges and package repositories as possible); (b) availability (i.e., it should be publicly available); (c) FOSS based (i.e., reflecting FOSS software development and artifacts); (d) history (i.e., the SBOM file should be part of the repository, statically embedded on its commit history). Consequentially, we have opted to search the Software Heritage Archive [20] (SWH). If a SBOM file is found in the public VCS of any of the main software forges, there is a high probability of being found in SWH, which archives over 4 billion commits and over 324 million diverse public software repositories, at the date of this writing [21]. Since there is no clear indication of which artifacts are SBOM files, we have defined a strategy to mine the archive, acquire possible candidates and then filter out non-SBOM files. The process is depicted in Figure 1 and each of the steps identified are detailed below.

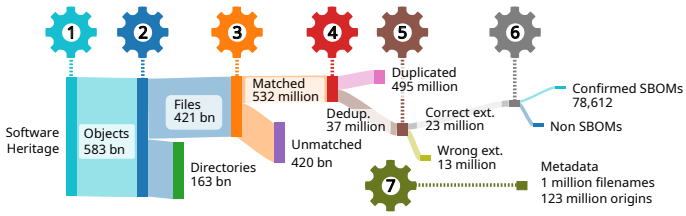

Fig. 1. Overview of the SBOM dataset creation

1) Download of indexes: First we download the *Optimized Row Columnar* (ORC) files that correspond to directory entries and the mapping of the intrinsic identifiers we will use (i.e., SHA1, and SWHID) for the release version "2024-08-23" of the Software Heritage graph dataset [22] on Amazon S3. The data contains 583,607,093,407 files and directory entries.

2) Filter for filenames: We then load those ORC files into Apache Spark [23] to retain only file entries. This results in 420,647,294,867 file references.

3) Search for regular expressions: The goal is to obtain as much candidates for SBOM files as possible. However, at such scale it is necessary to reduce the set of possible values. We use Spark to look for file names typically used to store SBOMs. Accordingly, we search any file name that contains, in a case insensitive way, at least one of the substrings: *spdx*, *swid*, *bom*, *cyclone*, *cdx*, or *dx*. This execution results in 531,729,785 items.

4) Deduplication of findings: The same SBOM content may be observed in different *commits* of the same repository, in different repositories, or with different file names. We save the list of file names to add to the metadata later (see section II-7). Then we drop duplicate contents (i.e., duplicate SHA1 hash values). This results in 37,036,596 candidates.

5) Removal of records with non-SBOM file extensions: We filter out records that contain file extensions which are known to belong to other than SBOM file formats (e.g., *.dll*,*.so* for dynamic libraries; *.odt*, for LibreOffice text documents; *.md* for markup documents). We obtain a list of 991 such extensions from the Apache mime-types file [24]. Then we obtain 23,590,858 records.

6) Download and evaluation of candidate SBOM files: We use the SHA1 hash values present in the candidate records to download the candidate files from Amazon S3, and them validate that their contents are not corrupted. Then we evaluate them to make sure to retain only real SBOM files. For this evaluation we have selected the *SBOM quality score (sbomqs)* tool [18], since it has already been used to score over 28 thousand SBOM files on the *sbombenchmark.dev* site [25]. We define that valid SBOM files for inclusion in the dataset are those that don't receive a *failed* result by the SBOM scoring tool. For those SBOMs included in the dataset, we store the results of the tool as metadata: quality score, SBOM standard, SBOM format and SBOM version. After the filtering process we obtain 78,612 confirmed SBOM files.

7) Metadata addition: We then select only those file names obtained in the deduplication step (section II-4) whose SHA1 hash values match the SHA1 hash values from the valid SBOM files. This results in 1,168,328 records with file names. We also traverse the SWH Graph to find all the repositories (i.e., *origins* in SWH) that have been observed for each SBOM file. The resulting data has 123,826,651 records with origins. Finally, we query SWH to obtain the earliest observed commit (i.e., *revision* in SWH) where the SBOM was observed, the earliest observed timestamp, and the total number of observed commits for each SBOM file. We sum all commits and observe that the SBOM files are distributed among 2,232,518,895 commits.

III. Description of the Dataset

The dataset contains a collection of SBOM files and the associated metadata, totalling 14 GB. We compress all elements of the dataset with *Zstandard* [26] to reduce the storage space, bringing it down to 5.6 GB.

SBOM files: The file *sbom-files.tar.zstd* contains all the 78,612 SBOM files. When extracted they occupy 12 GB of storage space. They are organized under a 1-level deep directory structure based on the first character of their SHA1 hash value, e.g., sbom-files/a/a01269419c76765dfa79499597e8eac79787450d. The files are deduplicated based on the SHA1 value (i.e., even if the same file was observed in different origins and different revisions, it will appear only once in the dataset).

Metadata: SBOM Metadata are provided as a set of 4 CSV [27] textual files, which correspond to 4 relational database tables, shown in Figure 2. The tables are described below, with the indication of the corresponding file:

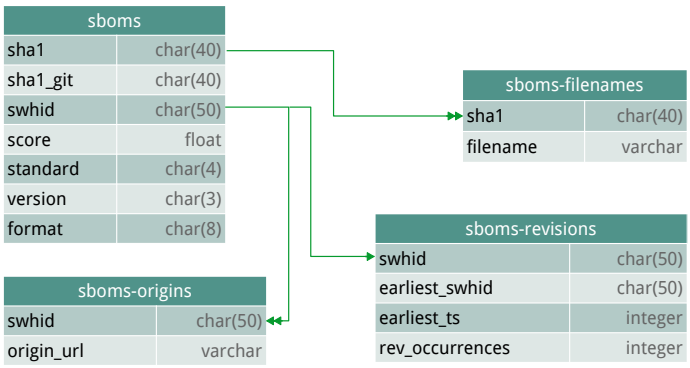

Fig. 2. Relational model for the SBOM dataset metadata

sboms (*sboms-01.csv.zst*) is the main table of the dataset and describes the main characteristics of each SBOM file. The *sha1* field is the SHA1 hash value of the content of the SBOM file, and its filename in the dataset; *sha1_git* is the original Git commit value where the SBOM was found; *swhid* is the file's Software Heritage persistent identifier (SWHID) [28]; *score* is the evaluation score given by *sbomqs* and ranges from 0 to 10; *standard* indicates the SBOM standard that is used for the SBOM: *spdx* (i.e., the SPDX stanard) or *cdx* (i.e. the CycloneDX standard); *format* indicates the layout format of the SBOM: *json*, *xml*, or *tag-value*.

sboms-filenames (*sboms-02-filenames.csv.zst*) provides the repository maintainer's choice for naming the SBOM, i.e., the file name that have been observed for each SBOM file. *sha1* is the SHA1 hash value for the SBOM contents and a foreign key to the *sboms* table; *filename* is an observed filename for that SBOM file, e.g., *camel-sbom.xml*, *bom.xml*. There can be many filenames observed for each SBOM content, i.e., many rows. A SBOM file which is renamed and committed to a VCS multiple times without content change will have one row for each distinct file name.

sboms-origins (*sboms-03-origins.csv.zst*) shows all the repositories where each SBOM file has been observed in. *swhid* is the file's SWHID and a foreign key to the *sboms* table; *oring_url* denotes the repository where the SBOM file was observed in the past, for example, *https://codeberg.org/cap_jmk/tinkabell.git*. There can be many repositories for each SBOM file, e.g. repository *forks*.

sboms-revisions (*sboms-04-revisions.csv.zst*) provides historical information. *earliest_swhid* gives the SWHID of the oldest known public commit that contained the SBOM file; *earliest_ts* is the commit timestamp as Unix time; *rev_occurrences* shows the total number of commits that contain the SBOM file, as known by SWH. For example, swh:1:cnt:0004b472fcd003bcbd29544a534d1f24afd0f3f2 denotes an instance of a SBOM for which the oldest commit is from May 8, 2022, and has a history of 52 revisions. By looking up the SHA1 value in *sboms* and consulting the file in *sbom-files* we can see it is a CycloneDX 1.3 JSON SBOM file for a Go language application.

IV. Dataset Use Cases

This dataset can be used to study SBOM creation practices in the wild, e.g., when they were created, main standards and formats, tools used, contents that are present. It can also be used to evaluate and benchmark SBOM related tools. Other possibilities include vulnerability analysis at large by searching the SBOM contents on vulnerability databases. We give some examples related to understanding SBOM practices below. Details for each of the examples are found in the *Jupyter Notebooks* provided with the dataset.

A. Standards

Since NTIA, who is pushing for SBOM adoption in the industry, has not endorsed any specific standards [7], it is up to the developers to decide what standard to adhere to. SWID Tags was proposed first and is now the standard ISO/IEC 19770-2:2015 [29]. SPDX is a project by the Linux Foundation and was standardized as ISO/IEC 5962:2021 [10]. CycloneDX is younger than both [11]. Do those elements affect general adoption? One first step to answering that is to verify what standards and file formats are really being used by developers.

We use the Python Pandas library [30] to load the *sboms* table and count the frequency of standards and formats found in the wild. Figure 3 show the results. This is a first approximation of the analysis because we are studying the *deduplicated* set only. Although a more comprehensive study might give some weights to SBOMs that have more origins and revisions, this first version considers the underlining choices of the original creators. It seems to be surprising that CycloneDX is the most used standard, while being the youngest proposal. Furthermore we see that there are no SWID Tags based SBOMs at all.

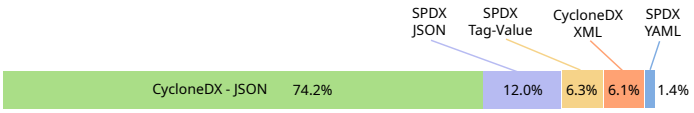

Fig. 3. Distribution of SBOM standards and file formats

B. Filenames

We haven't found in the SPDX specification guidelines for file extension or file naming conventions. OpenSSF Best Practices document recommends to add the extension **.spdx* for tag-value format or **.spdx.{json, xml, yml, yaml, rdf, rdf-xml}* [31]. CycloneDX recommends the adoption of the filenames *bom.{json, xml}* or the extensions **.cdx.{json, xml}* [32]. We will see what practitioners adhere to.

We use Pandas to load the table *sboms-filenames*, count the number of occurrences and display the 10 most used file names. Figure 4. For this example, we look at all filenames that are used in all repositories throughout all commits.

C. Popular forges and package repositories

The dataset contains 94,618,356 unique origins. Each of those origins point to one of 1,782 unique forges. We examine

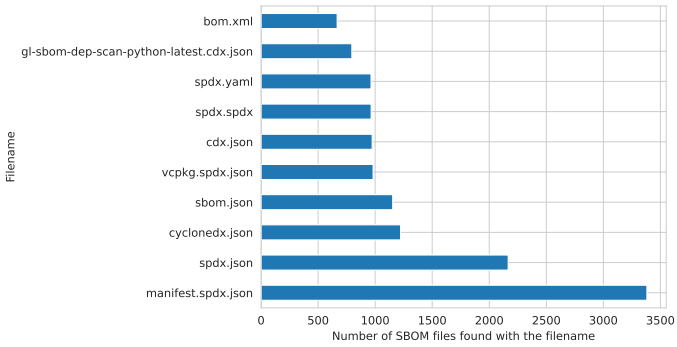

Fig. 4. The most popular filenames for SBOM files

how SBOM practitioners' works are distributed along the forges.

We use Pandas to load the table *sboms-origins* and remove the duplicates of *origin_url*. Then we use the library *tldextract* [33] to help extract the part that correspond to the forge. In total, the practitioners have placed SBOMs in 1,783 unique forges. The top ten forges that contributed with the most SBOM files to the dataset are shown in figure 5.

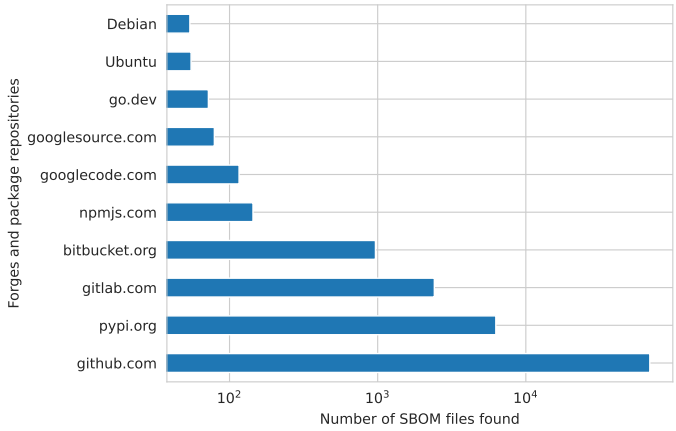

Fig. 5. Forges with the most SBOM files

V. Limitations

We took the approach of searching for SBOM files that are stored in public VCS. However, most of the forges also have a way (e.g., Github assets) for the contributors to release software artifacts that are stored outside the version control system and SBOM files could be placed there as well. Additionally, the popular forges Github [34] and Gitlab [35] have recently announced services to integrate dynamic SBOM generation in their pipelines. Our approach in this work doesn't account for those SBOM files, as they are released outside of VCS repositories. There is still no consensus to where SBOM files should be stored, and it is possible to argue for storing them on either place. If practitioners favor the approach that requires the least amount of work, they may use the platform's SBOM creation functionalities, and fewer SBOM files will be found in public VCS repositories. However, storing SBOM files inside the VCS repositories makes them independent of that specific forge (i.e., cloning the VCS will retrieve all resources). Furthermore, we sampled some repositories from the dataset and looked at their VCS contents in SWH. There are contributors that include the SBOM file in the same directory of the release notes, inside the VCS, where they are kept as part of the history along with the source files.

VI. Related Work

Previous works have created SBOM datasets from existing projects, as part of a broader research to evaluate SBOM generating tools for feature comparison [36], on Docker images [37], for security purposes [38], or for accuracy assessment [15], [8], [39], [40]. There are two main differences that our work presents. First, our work doesn't generate the SBOM files. We aggregate those committed by practitioners to VCS repositories, which possibly provides for more diversity in the conditions of generation (e.g., time of creation, origin, authors, generation tools) to better be able to assess SBOM practices; Second, while the number of SBOM files in those works range from 2 to 7,876, our dataset is much larger by an order of magnitude.

There are also related works that gather SBOM files in the wild. Ambala has mined SBOM files from 18 software repositories for security analysis [41]. The dataset "bom-shelter" contains over 50 SBOM files found by using a web site specialized on source code search [42]. Nocera et al have selected SBOM creation tools that have Github repositories and then they have used Github's API to locate software repositories that depend on those tools, finding 186 SBOM files in the wild [43]. O'Donoghue has obtained 1,151 SBOM files from Interlynk's SBOM sbombenchmark.dev [25], by accessing directly its Amazon S3 repository, to check them for vulnerabilities using SBOM tools [44]. Our approach differs from these works on the diversity and quantity of SBOM files found in the wild. While most studies rely on sourcing one or a few well known software forges and collections (e.g., GitHub, sbombenchmark.dev), our work represents 1,782 unique forges and 94,618,356 unique source repositories, providing a larger and a much more diverse dataset.

VII. Conclusion

We have introduced a dataset of 78,612 unique SBOM files found in the wild. They were observed in 94,618,356 repositories, spread in 1,782 unique forges. We also included the metadata: *sha1*, *sha1_git*, *swhid*, SBOM score, SBOM standard, SBOM version, SBOM format, and provenance information. We've given some suggestions of future usage and we have presented some concrete examples.

Future work: The goal is to enlarge the dataset and to improve the available metadata. We will include newer SBOMs found in the wild and also add metadata related to quality criteria as measured by diverse SBOM tools.

References

[1] P. Buchkova, J. H. Hinnerskov, K. Olsen, and R.-H. Pfeiffer, "DaSEA," in *Proceedings of the 19th International Conference on Mining Software Repositories*, ACM, may 2022.

[2] A. Decan, T. Mens, and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems," *Empirical Software Engineering*, vol. 24, pp. 381–416, feb 2018.

[3] M. Hoffmann, F. Nagle, and Y. Zhou, "The value of open source software," *SSRN Electronic Journal*, 2024.

[4] S. Phipps and S. Zacchiroli, "Continuous open source license compliance," *Computer*, vol. 53, pp. 115–119, dec 2020.

[5] N. Harutyunyan and D. Riehle, "Industry best practices for open source governance and component reuse," in *Proceedings of the 24th European Conference on Pattern Languages of Programs*, ACM, jul 2019.

[6] Y.-K. Lin and W. Li, "A theory of open source security: The spillover of security knowledge in vulnerability disclosures through software supply chains," *SSRN Electronic Journal*, 2023.

[7] T. Stalnaker, N. Wintersgill, O. Chaparro, M. Di Penta, D. M. German, and D. Poshyvanyk, "Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, pp. 1–13, ACM, Feb. 2024.

[8] M. Mirakhorli, D. Garcia, S. Dillon, K. Laporte, M. Morrison, H. Lu, V. Koscinski, and C. Enoch, "A landscape study of open source and proprietary tools for software bill of materials (sbom)," 2024.

[9] D. Waltermire, B. A. Cheikes, L. Feldman, and G. Witte, *Guidelines for the Creation of Interoperable Software Identification (SWID) Tags*. Apr. 2016.

[10] "About spdx." https://spdx.dev/about/overview/. Accessed: 2024-11-11.

[11] "Cyclonedx history." https://cyclonedx.org/about/history/. Accessed: 2024-11-11.

[12] 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, pp. 82–88, mar 2023.

[13] M. Dalla Preda, S. Egelman, A. M. Mandalari, V. Stocker, J. Tapiador, and N. Vallina-Rodriguez, "Eu cyber resilience act: Socio-technical and research challenges (dagstuhl seminar 24112)," 2024.

[14] W. Otoda, T. Kanda, Y. Manabe, K. Inoue, and Y. Higo, "Sbom challenges for developers: From analysis of stack overflow questions," in *2024 IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA)*, vol. 10, pp. 43–46, IEEE, May 2024.

[15] M. Balliu, B. Baudry, S. Bobadilla, M. Ekstedt, M. Monperrus, J. Ron, A. Sharma, G. Skoglund, C. Soto-Valero, and M. Wittlinger, "Challenges of producing software bill of materials for java," arXiv, 2023.

[16] "The state of software bill of materials (sbom) and cybersecurity readiness." https://www.linuxfoundation.org/research/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness. Accessed: 2024-11-02.

[17] S. Torres-Arias, D. Geer, and J. S. Meyers, "A viewpoint on knowing software: Bill of materials quality when you see it," *IEEE Security & Privacy*, vol. 21, pp. 50–54, Nov. 2023.

[18] "Sbom quality score - quality metrics for your sboms." https://github.com/interlynk-io/sbomqs. Accessed: 2024-11-11.

[19] L. Soeiro, T. Robert, and S. Zacchiroli, "Replication package for wild SBOMs: a large-scale dataset of software bills of materials from public code." https://doi.org/10.5281/zenodo.14250102, 2024.

[20] R. Di Cosmo and S. Zacchiroli, "Software heritage: Why and how to preserve software source code," in *iPRES 2017 - 14th International Conference on Digital Preservation*, (Kyoto, Japan), pp. 1–10, Sept. 2017.

[21] "Software heritage archive." https://archive.softwareheritage.org/. Accessed: 2024-10-02.

[22] A. Pietri, D. Spinellis, and S. Zacchiroli, "The software heritage graph dataset: Public software development under one roof," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, IEEE, may 2019.

[23] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, pp. 145–164, Oct. 2016.

[24] "Apache httpd mime.types." https://svn.apache.org/viewvc?view= revision&evision=1301894. Accessed: 2024-06-15.

[25] "Sbom benchmark." https://sbombenchmark.dev/. Accessed: 2024-11-11.

[26] Y. Collet and M. Kucherawy, "Zstandard Compression and the 'application/zstd' Media Type." RFC 8878, Feb. 2021. Accessed: 2024-11-02.

[27] Y. Shafranovich, "Common Format and MIME Type for Comma-Separated Values (CSV) Files." RFC 4180, Oct. 2005.

[28] R. Di Cosmo, M. Gruenpeter, and S. Zacchiroli, "Identifiers for digital objects: the case of software source code preservation," in *iPRES 2018-15th International Conference on Digital Preservation*, pp. 1–9, 2018.

[29] "Software identification (swid) tagging." https://csrc.nist.gov/Projects/Software-Identification-SWID. Accessed: 2024-11-26.

[30] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[31] "Best practices for naming and directory conventions for sboms (software bill of materials) in open source projects." http://sbom-catalog.openssf.org/sbom-naming.html. Accessed: 2024-11-26.

[32] "Cyclonedx specification overview." https://cyclonedx.org/specification/overview/. Accessed: 2024-11-26.

[33] J. Kurkowski, "Tldextract." https://pypi. org/project/tldextract, 2019.

[34] "Introducing self-service sboms." https://github.blog/enterprise-software/governance-and-compliance/introducing-self-service-sboms/. Accessed: 2024-11-03.

[35] "The ultimate guide to sboms." https://about.gitlab.com/blog/2022/10/25/the-ultimate-guide-to-sboms/#gitlab-and-dynamic-sboms. Accessed: 2024-11-03.

[36] G. Dalia, C. A. Visaggio, A. Di Sorbo, and G. Canfora, "Sbom ouverture: What we need and what we have," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ARES 2024, pp. 1–9, ACM, July 2024.

[37] N. Kawaguchi, C. Hart, and H. Uchiyama, "Understanding the effectiveness of sbom generation tools for manually installed packages in docker containers," *Journal of Internet Services and Information Security*, vol. 14, pp. 191–212, Aug. 2024.

[38] O. Kagizmandere and H. Arslan, "Vulnerability analysis based on sboms: A model proposal for automated vulnerability scanning for ci/cd pipelines," *International Journal of Information Security Science*, vol. 13, pp. 33–42, June 2024.

[39] S. Yu, W. Song, X. Hu, and H. Yin, "On the correctness of metadata-based sbom generation: A differential analysis approach," in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 29–36, IEEE, June 2024.

[40] M. F. Rabbi, A. I. Champa, C. Nachuma, and M. F. Zibran, "Sbom generation tools under microscope: A focus on the npm ecosystem," in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, pp. 1233–1241, ACM, Apr. 2024.

[41] A. Ambala, "Exploring the dynamics of software bill of materials (sboms) and security integration in open source projects." https://urn.kb.se/resolve?urn=urn:nbn:se:bth-26057, 2024.

[42] "Are SBOMs any good? Preliminary measurement of the quality of open source project SBOMs." https://www.chainguard.dev/unchained/are-sboms-any-good-preliminary-measurement-of-the-quality-of-open-source-project-sboms, 2022. Accessed: 2024-11-02.

[43] S. Nocera, S. Romano, M. D. Penta, R. Francese, and G. Scanniello, "Software bill of materials adoption: A mining study from github," in *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, Oct. 2023.

[44] E. J. O'Donoghue, "Using software bill of materials for software supply chain security and its generation impact on vulnerability detection," 2024.