Probing Challenges and Future Research of SBOM Generation for Medical Devices

Hui Zhuang*§, Yan Long*, Duyeong Kim†, Jennifer R. Amos‡, Heejo Lee†, Kevin Fu*
* *Northeastern University, Boston, USA*
† *Korea University, Seoul, Republic of Korea*
‡ *University of Illinois Urbana-Champaign, Urbana, USA*
{*zhuang.hu, y.long, k.fu*}*@northeastern.edu*, {*duyeong, heejo*}*@korea.ac.kr, jamos@illinois.edu*

Abstract—**Medical devices increasingly incorporate third-party software components, which have prompted the U.S. Food and Drug Administration (FDA) to require manufacturers to submit a Software Bill of Materials (SBOM) during the pre-market phase for cybersecurity management. This short paper investigates to what degree existing SBOM generation tools meet regulatory requirements. We perform a case study on a widely used open-source SBOM tool (Anchore Syft) to generate and analyze SBOMs for Tidepool Loop, an FDA-approved software kit. Preliminary experimental results reveal deficiencies in the generated SBOMs such as missing component relationships, unknown version, and unspecified supplier names, indicating potential problems in data completeness and accuracy in broader landscapes of SBOM generation for medical devices. We further discuss important future research topics, including collaborative efforts among multiple stakeholders, practitioner-oriented SBOM surveys, and benchmarks for evaluating SBOM quality.**

1. Introduction

Medical devices are increasingly integrating a variety of third-party components [1]. While these third-party components improve the functionality and development efficiency of medical devices, they also increase the risk of cybersecurity vulnerabilities, as demonstrated by well-known vulnerabilities such as Log4j [2] and WannaCry [3]. To address these risks, the U.S. Food and Drug Administration (FDA) [4] requires medical device manufacturers to submit a Software Bill of Materials (SBOM) during the pre-market stage, enabling manufacturers and regulatory agencies to verify whether components are exposed to CISA-listed KEV vulnerabilities [5], and to promptly identify devices impacted by newly discovered vulnerabilities, which is critical given the severe consequences of cyberattacks on healthcare systems [6]. Given that manual compilation of SBOMs is often impractical due to the high complexity of medical devices and the widespread integration of third-party components, automated SBOM generation tools are widely used to identify components and improve generation efficiency.

However, the dependency of SBOM's accuracy on automated generation tools prompts us to think about an important emerging problem: **To what extent could existing SBOM generation tools meet regulatory expectations in the medical device domain?** As FDA stipulates that medical device manufacturers should submit SBOMs that adhere to the minimum elements (Table A.1) defined by the National Telecommunications and Information Administration (NTIA) [7], the first step to evaluate whether existing SBOM generation tools meet FDA expectations involves assessing whether the generated SBOMs comprehensively and accurately capture these critical elements.

Although a variety of SBOM generation tools have already been deployed in the medical device industry [8]–[11], their commercial nature makes it difficult to assess their detailed technical capability and implementation issues, preventing researchers from conducting systematic analysis to examine potential flaws. As the first step of probing the challenges in SBOM generation for medical devices, this work conducted a preliminary test using open-source generation tools and medical device software to evaluate the potential problems of existing SBOM generation schemes in the medical device domain. Specifically, we used the Anchore Syft [12] to generate an SBOM for the open-source medical device application Tidepool Loop [13], which is the first FDA-approved diabetes-related app [14], and compared the generated SBOM with current regulatory requirements. The results revealed that the SBOM generated by Syft exhibited several missing key elements, including (1) missing relationships, (2) unknown component versions, and (3) unspecified supplier names.

Our analysis shows that these problems are caused by limitations in tool implementation, variability in dependency specification strategies, and the lack of standardized metadata support in programming language ecosystems. Many of the observed challenges stem from broader technical limitations not exclusive to Syft or open-source tools. Therefore, we argue that commercial SBOM generation tools are likely to also encounter similar challenges in SBOM data compliance. In addition, we found that existing SBOM evaluation tools do not fully capture the issues identified in our analysis when assessing the quality of generated SBOMs. We further discuss the plan of conducting a survey with SBOM professionals working in the medical device security industry. Aiming to characterize the potential gaps between open-source and commercial SBOM generation software.

§*Corresponding author: zhuang.hu@northeastern.edu*

Furthermore, we discuss the development benchmarks that could enable consistent evaluation across tools and support measurable improvement in both data completeness and accuracy. We show how coordinated efforts among regulatory agencies, medical device manufacturers, SBOM tool developers, and upstream software providers are urgently needed to jointly improve the quality and regulatory compliance of SBOMs.

2. FDA's SBOM Requirement

Since 2023, the U.S. FDA requires manufacturers to submit a SBOM during the pre-market phase to address the growing cybersecurity threats to medical devices [4].

Although the FDA requires SBOM submission, it has not established a formal specification for the structure or required fields. Instead, the FDA advises manufacturers to adopt the SBOM standard proposed by NTIA [7], which specifies that an SBOM must include, at a minimum, the seven data elements listed in Table A.1, with detailed descriptions provided in the Description column. These elements help both device manufacturers and healthcare providers track vulnerabilities and fix problems quickly when they arise. We emphasize that this table reflects only the minimal requirements, SBOMs are typically more detailed like Figure A.1 in practice.

To meet this requirement, manufacturers in the medical device industry typically rely on automated tools to generate SBOMs, as manually documenting a large number of third-party components is impractical. While such tools have been widely adopted in other areas of software engineering, recent studies suggest that their accuracy remains limited. For instance, Rabbi et al. [15] found that tools such as Syft achieve only around 80% accuracy in extracting critical metadata and most tools fail to automatically detect inter-component dependencies, resulting in SBOMs with missing or erroneous fields. Although their work focused on JavaScript npm projects rather than medical device software, the third-party components used in medical devices are largely similar to those in other applications and systems, we speculate that these known issues may also affect SBOM generation in the medical device domain, limiting the ability of regulatory compliance.

3. Preliminary Experiment & Observations

3.1. Problems of SBOM Generation Tools

To investigate whether existing SBOM generation tools also encounter extraction accuracy issues in the medical device domain, we employed the open-source tool Syft [12] to generate an SBOM for the Tidepool Loop application [13]. Tidepool Loop is an insulin delivery app that runs on the iPhone and is the first FDA-approved application for diabetes management [14]. SBOMs can be generated through various approaches, such as binary-based [16]–[20], filesystem-based [12], [21], [22], and source-based [23]–[25] analysis. We used Syft to generate SBOMs through its default filesystem-based analysis, which is more general and does not require source code or build manifests, an example of which is shown in Figure A.1. Our analysis of the generation SBOMs identified three major issues:

Missing Component Relationship: Component relationship information is essential for revealing vulnerability propagation paths and for guiding precise remediation. The completeness of component relationship information in the SBOM was observed to depend on the selected output format. When using the SPDX format, Syft can automatically include inter-component relationships. However, the resulting SBOM lacks any relationship data when the output is set to CycloneDX. By examining discussions in Syft's GitHub issues and comparing SBOMs generated by another tool cdxgen [26], we found that Syft omits component relationships in CycloneDX SBOMs due to limitations in its processing logic for this format. This was confirmed by generating a CycloneDX SBOM for Tidepool Loop using cdxgen, which included complete inter-component relationship information, showing that the issue stems from Syft itself rather than from the CycloneDX format. This finding demonstrates that the quality of an SBOM largely depends on the tool selected, highlighting the need for a standardized evaluation benchmark in the medical device domain to systematically assess the performance of different SBOM tools.

Unknown Version: Accurate version information is critical for linking SBOM components to vulnerability databases. The SBOM generated by Syft for the Loop application reveals that approximately 14% of its components (15 out of 107) lack version information. This issue primarily stems from how developers specify dependencies. For example, in this Swift-based application, dependencies are listed in the Package.resolved file. We observed that the "state" field in different packages varies: some include a version field, which Syft [12] can successfully extract; others only include a branch or revision, which results in the version being marked as "unknown" in the SBOM. This discrepancy originates from the dependency specification strategy adopted by developers. When a fixed version is declared, the Package.resolved file contains a corresponding version field. In contrast, if a dependency is defined using a branch (e.g., "main") or a specific commit, only the branch or revision is recorded, and no standardized version field is provided. As a result, Syft is unable to extract explicit version information from such entries. To mitigate this issue, in addition to encouraging developers to specify concrete version numbers in dependency declarations, it is also crucial to enhance SBOM tools to extract branch or commit information when explicit versions are unavailable for improving the completeness and traceability of SBOM outputs.

Unspecified Supplier Name: Supplier information is important for regulatory accountability and identification of the organization responsible for addressing vulnerabilities. However, we found that almost all components have the problem of missing supplier names in the generated SBOMs. The absence of supplier name fields is attributable to two main reasons: first, most programming language ecosystems do not enforce supplier declaration; second, Syft lacks robust support for extracting supplier names from the package URL (purl). We compared SBOMs generated by Syft in both CycloneDX and SPDX formats and found that the CycloneDX format entirely omits the supplier name field. Although the SPDX format does include this field, whether it is successfully populated with meaningful supplier information depends on the component type, which can typically be inferred from the prefix following "pkg:" in the purl. For example, when the component type is github, Syft is able to correctly extract the supplier name. However, for components of type swift and gem, the supplier field consistently remains empty. This issue is partially due to the fact that neither the Swift nor RubyGems ecosystems require developers to explicitly specify supplier information. More importantly, we observed that the purl of Swift components actually includes supplier information in the segment immediately preceding the component name. Nevertheless, Syft still fails to parse it correctly, indicating that its current handling of purl structure remains incomplete and requires further improvement. This result indicates that the quality of SBOM generation is influenced not only by the parsing capabilities of tools but also by the metadata specification of programming language ecosystems. For SBOM tool developers, it is important to enhance support for parsing various purl formats. In addition, we encourage programming language maintainers to promote more structured metadata specifications.

3.2. Problems of SBOM Evaluation Tools

Beyond evaluating SBOM generation tools, we also investigated the performance of representative SBOM evaluation tools to test their effectiveness in assessing SBOM quality. Existing SBOM evaluation solutions include tools such as sbomqs [27] and CycloneDX CLI [28]. We take sbomqs as an example. The experiments focus on the "NTIA-minimum-elements" metric in sbomqs, which evaluates whether an SBOM satisfies the minimum data element requirements defined by the NTIA. Based on this metric, we evaluate CycloneDX and SPDX SBOMs generated by Syft for the Tidepool Loop application, with results shown in Table A.2. The scores on some features such as sbom_dependencies, comp_with_supplier were generally consistent with our earlier observations.

However, we further identified limitations in sbomqs's evaluation logic. For instance, while the comp_with_version field is mostly marked as correct, our manual analysis reveals that sbomqs assigns a score to this field as long as the version field value exists, even if it is unknown. This shows that sbomqs evaluates certain features mainly by checking whether field values exist, but it does not assess their accuracy. Moreover, we observed that even for the same application, sbomqs calculates the number of evaluated components differently depending on the SBOM format: in CycloneDX format, it scores both components and files, whereas in SPDX format, it only scores components. The discrepancy arises from differences in structural modeling: CycloneDX consolidates third-party components and files under the component structure, while SPDX treats them as parallel constructs. However, sbomqs does not granularly account for this distinction and instead performs coarse-grained extraction of the entire structure containing third-party components, leading to inconsistencies in the reported component count across SBOM formats. It is possible to imagine that if a regulatory agency uses tools similar to sbomqs to evaluate the quality of submitted SBOMs, the conclusion will inevitably be biased. As a result, we argue that improving overall cybersecurity practices using SBOMs requires improving both SBOM generation and evaluation capabilities.

4. Future Research Directions

4.1. SBOM Quality Improvement

Improving SBOM quality requires collaboration across multiple stakeholders including regulatory agencies, medical device manufacturers, SBOM tool developers, and upstream software providers, as the challenges stem not only from technical limitations of current tools but also from organizational and process-related factors. While the intent of the regulation is to enhance transparency and mitigate risks from software vulnerabilities, the regulation also introduces new complexities for manufacturers. However, there remains a significant lack of empirical evidence detailing the specific challenges that medical device manufacturers face in complying with this requirement, making it difficult to tailor support or policy interventions effectively. Moreover, SBOMs are often developed by cybersecurity professionals who may not fully understand the workflows of medical device manufacturers or the realities of clinical use environments, leading to potential misalignments in implementation. Despite these hurdles, the regulation also presents an opportunity: by fostering cross-disciplinary collaboration, investing in workforce training, and aligning SBOM practices with real-world manufacturing and clinical contexts, stakeholders can not only meet regulatory expectations but also advance the safety and resilience of medical technologies. To realize this potential, each stakeholder group must take coordinated and well-defined actions.

Regulatory agencies should develop SBOM generation standards and formats specifically for medical devices, urging manufacturers to adhere to them during both the development and maintenance phases. It would also be beneficial if they could establish unified benchmarks for SBOM evaluation, providing clear directions for tool developers and reliable criteria for manufacturers in selecting tools.

Medical device manufacturers should recognize the critical role of SBOMs across the entire product lifecycle. When incorporating third-party components, they are encouraged to prioritize components that meet SBOM compliance requirements. When deficiencies are identified, manufacturers should bear the ultimate responsibility for correcting and maintaining SBOM accuracy. Additionally, manufacturers should be capable of identifying practical challenges encountered during SBOM deployment in the medical device domain, such as data update complexity, regulatory audit procedures, and compatibility within clinical environments, in order to provide valuable feedback for compliance improvements and cross-sector coordination.

SBOM tool developers should enhance data field parsing logic and improve the tool's capacity to automatically complete component metadata. By integrating metadata and external databases, tools can perform intelligent inference to compensate for missing fields caused by nonstandard upstream data. In addition, to address the diversity of SBOM formats, developers should design differentiated parsing strategies for different formats to ensure that SBOMs generated in various formats meet regulatory requirements for both compliance and accuracy. Moreover, developers are encouraged to actively engage with practitioners in the medical device industry to better understand domain-specific SBOM needs and usage contexts.

Finally, we believe that upstream software providers can contribute by promoting the inclusion of essential metadata such as version numbers and supplier identities in development specifications to meet the NTIA's minimum elements requirements. This approach will not only improve the accuracy of data extraction by SBOM tools but also strengthen the overall quality and traceability of SBOMs.

4.2. Practitioner Survey

We believe a survey and interview study is necessary for further addressing the gap between speculated problems and real-world usage patterns. In Section 3.1, we highlighted several issues observed in existing open-source SBOM generation tools. However, it remains unclear which specific commercial or open-source tools are actually used by medical device manufacturers, and whether these tools exhibit similar or new types of issues. Therefore, empirical studies are essential to bridge the gap between hypothesized concerns and practical usage in real-world settings.

The interviews and surveys will target industry professionals working in the medical device domain and emphasize practical challenges of SBOM generation in the wild, with a focus on investigating answers to the following research questions:

- What are the unique challenges of SBOM generation for medical and healthcare devices, compared to other application domains?
- How effectively do existing SBOM generation tools support the unique needs of medical devices?
- How do manufacturers keep SBOMs up to date with software changes?
- How to verify that a generated SBOM accurately represents all third-party components in a medical device?
- What kind of knowledge or guidance do medical device manufacturers want to receive from the FDA?
- How can the community collaborate to develop SBOM standards better suited for medical devices?

4.3. Medical Device SBOM Benchmark

Establishing a benchmark for SBOM generation tools is another necessary step toward standardized evaluation and cross-tool comparison. We envision creating a representative dataset of medical device samples, generating SBOMs using various tools, and systematically evaluating them based on a common set of metrics.

The primary challenge in building such a benchmark lies in obtaining reliable ground truth. As discussed in Section 3.2, existing SBOM evaluation tools have limitations in verifying data accuracy. Therefore, high-quality ground truth is urgently needed to assess the output of various SBOM tools. However, commercial medical devices typically contain numerous third-party components, making manual construction of complete and trustworthy ground truth infeasible.

In future research, efforts can be directed toward two key areas. First, collaboration between regulatory agencies and medical device manufacturers could facilitate the creation of standardized datasets containing approved products and their associated SBOMs. These datasets can serve as trusted references for advancing evaluation technologies tailored to SBOM generation in the medical device domain. Second, the integration of large language models (LLMs) into SBOM evaluation processes presents a promising direction. For instance, LLMs are potentially capable of automatically identifying and extracting SBOM elements and verifying the accuracy of field values through component information retrieval, thereby enhancing the effectiveness of automated SBOM assessments.

5. Conclusion

This paper focuses on SBOM generation in the medical device domain and analyzes the limitations of the open-source SBOM generation tool in producing SBOMs for medical devices. Based on preliminary experimental results, the study identifies two key directions for future research in this area: engaging with stakeholders to better understand their use of commercial SBOM generation tools and developing a unified evaluation benchmark to enable assessment and comparison of different SBOM tools.

Acknowledgment

We are grateful to all anonymous reviewers for their valuable feedback that helped improve this paper, and we thank Christopher Pellegrini for his constructive advice. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00440780, Development of Automated SBOM and VEX Verification Technologies for Securing Software Supply Chains, No.RS-2022-II220277, Development of SBOM Technologies for Securing Software Supply Chains).

References

[1] 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. 1, pp. 1–6, 2021.

[2] UK National Cyber Security Centre. (2021) Log4j vulnerability – what everyone needs to know. Accessed: 2025-07-22. [Online]. Available: https://www.ncsc.gov.uk/information/log4j-vulnerability-what-everyone-needs-to-know

[3] Wikipedia contributors, “Wannacry ransomware attack,” https://en.wikipedia.org/wiki/WannaCry_ransomware_attack, 2024, accessed: 2025-07-22.

[4] U.S. Food and Drug Administration, “Cybersecurity in medical devices: Quality system considerations and content of premarket submissions,” https://www.fda.gov/regulatory-information/search-fda-guidance-documents/cybersecurity-medical-devices-quality-system-considerations-and-content-premarket-submissions, Jun. 2025, guidance for Industry and Food and Drug Administration Staff, Docket No. FDA-2021-D-1158. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/cybersecurity-medical-devices-quality-system-considerations-and-content-premarket-submissions

[5] Cybersecurity and Infrastructure Security Agency. (2024) Known exploited vulnerabilities catalog. Accessed: July 23, 2025. [Online]. Available: https://www.cisa.gov/known-exploited-vulnerabilities-catalog

[6] N. Sullivan and K. Raphel, “Clinical and hospital system emergency management implications of cyberthreats,” in *Proceedings of the 2024 Workshop on Cybersecurity in Healthcare*, ser. HealthSec ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 11–16. [Online]. Available: https://doi.org/10.1145/3689942.3694742

[7] NTIA Multistakeholder Process on Software Component Transparency Framing Working Group, “Framing software component transparency: Establishing a common software bill of materials (sbom), second edition,” https://www.ntia.gov/sites/default/files/publications/ntia_sbom_framing_2nd_edition_20211021_0.pdf, Oct. 2021, accessed 2025-07-09.

[8] Blue Goat Cyber, “Fda-compliant sbom generation & maintenance for medical devices,” https://bluegoatcyber.com/services/sbom-software-bill-of-materials-services/, 2025, accessed July 2025.

[9] FOSSA, “Sbom generation and management for medical devices,” https://fossa.com/industries/medical-device/, 2025, accessed July 2025.

[10] Ketryx, “Medical device sbom management software,” https://www.ketryx.com/capabilities/sbom-software-bill-of-materials, 2025, accessed July 2025.

[11] Labrador Labs, “Secure software supply chain management,” https://labradorlabs.ai, 2025, accessed: July 23, 2025.

[12] I. Anchore, “Syft: Cli tool and library for generating a software bill of materials from container images and filesystems,” https://github.com/anchore/syft, 2020, version 1.x.

[13] Tidepool. (2025) Tidepool loop receives fda clearance. Accessed: 2025-07-13. [Online]. Available: https://www.tidepool.org/tidepool-loop

[14] H. Look, “Tidepool loop has received fda clearance!” https://tidepool.org/blog/tidepool-loop-has-received-fda-clearance, January 2023, accessed July 2025.

[15] 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*, ser. SAC ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1233–1241. [Online]. Available: https://doi.org/10.1145/3605098.3635927

[16] W. Tang, P. Luo, J. Fu, and D. Zhang, “Libdx: A cross-platform and accurate system to detect third-party libraries in binary code,” in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020, pp. 104–115.

[17] C. Yang, Z. Xu, H. Chen, Y. Liu, X. Gong, and B. Liu, “Modx: binary level partially imported third-party library detection via program modularization and semantic matching,” in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1393–1405. [Online]. Available: https://doi.org/10.1145/3510003.3510627

[18] S. Li, Y. Wang, C. Dong, S. Yang, H. Li, H. Sun, Z. Lang, Z. Chen, W. Wang, H. Zhu, and L. Sun, “Libam: An area matching framework for detecting third-party libraries in binaries,” *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 2, Dec. 2023. [Online]. Available: https://doi.org/10.1145/3625294

[19] L. Jiang, J. An, H. Huang, Q. Tang, S. Nie, S. Wu, and Y. Zhang, “ BinaryAI: Binary Software Composition Analysis via Intelligent Binary Source Code Matching ,” in *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2024, pp. 2771–2783. [Online]. Available: https://doi.ieeecomputersociety.org/10.1145/3597503.3639100

[20] Z. Yuan, M. Feng, F. Li, G. Ban, Y. Xiao, S. Wang, Q. Tang, H. Su, C. Yu, J. Xu, A. Piao, J. Xuey, and W. Huo, “B2sfinder: Detecting open-source software reuse in cots software,” in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 1038–1049.

[21] A. Security, “Trivy: Vulnerability and misconfiguration scanner,” https://github.com/aquasecurity/trivy, 2024, accessed: July 24, 2025.

[22] I. VMware, “Tern: A software composition analysis tool for containers,” https://github.com/tern-tools/tern, 2024, accessed: July 24, 2025.

[23] S. Woo, S. Park, S. Kim, H. Lee, and H. Oh, “Centris: A precise and scalable approach for identifying modified open-source software reuse,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 860–872.

[24] J. Wu, Z. Xu, W. Tang, L. Zhang, Y. Wu, C. Liu, K. Sun, L. Zhao, and Y. Liu, “Ossfp: Precise and scalable c/c++ third-party library detection using fingerprinting functions,” in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 270–282.

[25] Y. Na, S. Woo, J. Lee, and H. Lee, “Cneps: A precise approach for examining dependencies among third-party c/c++ open-source components,” in *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, 2024, pp. 2918–2929.

[26] S. Springett and C. C. Team, “cdxgen: A universal SBOM generator for multiple ecosystems,” https://github.com/CycloneDX/cdxgen, CycloneDX Project, 2023, version 9.x or later. Accessed July 14, 2025.

[27] Interlynk-io, “sbomqs: Quality metrics for sboms,” https://github.com/interlynk-io/sbomqs, 2025, accessed: 2025-07-21.

[28] O. Foundation, “Cyclonedx cli: Sbom analysis and conversion tool,” https://github.com/CycloneDX/cyclonedx-cli, 2025, accessed: 2025-07-21.

Appendix

1. NTIA-Specified Minimum Elements for SBOM

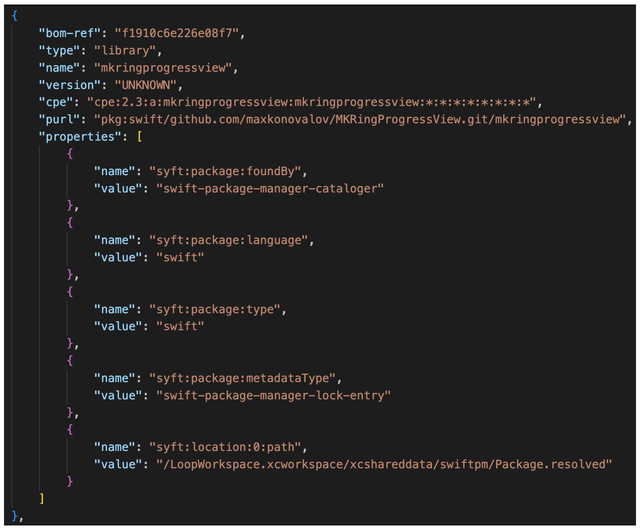
Table A.1 outlines the NTIA-specified minimum elements for an SBOM and provides a description of each element.

Attributes	Description
Author Name	Author of the SBOM
Timestamp	Date and time when the SBOM was last updated
Supplier Name	Name or other identifier of a component supplier
Component Name	Name or other identifier of a component
Version String	Version of a component
Unique Identifier	Information to uniquely define a component
Relationship	Association between SBOM components

TABLE A.1: NTIA-Specified Minimum Elements for SBOM [7]

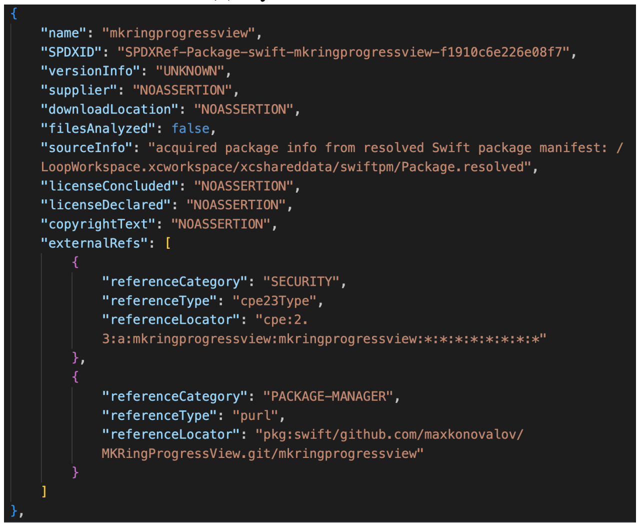
2. Example SBOM Output

Figure A.1 shows SBOM component information for the Tidepool Loop application generated using the Syft tool in both CycloneDX and SPDX formats.



```
{
    "bom-ref": "f1910c6e226e08f7",
    "type": "library",
    "name": "mkringprogressview",
    "version": "UNKNOWN",
    "cpe": "cpe:2.3:a:mkringprogressview:mkringprogressview:*:*:*:*:*:*:*:*",
    "purl": "pkg:swift/github.com/maxkonovalov/MKRingProgressView.git/mkringprogressview",
    "properties": [
        {
            "name": "syft:package:foundBy",
            "value": "swift-package-manager-cataloger"
        },
        {
            "name": "syft:package:language",
            "value": "swift"
        },
        {
            "name": "syft:package:type",
            "value": "swift"
        },
        {
            "name": "syft:package:metadataType",
            "value": "swift-package-manager-lock-entry"
        },
        {
            "name": "syft:location:0:path",
            "value": "/LoopWorkspace.xcworkspace/xcshareddata/swiftpm/Package.resolved"
        }
    ]
},
```

(a) CycloneDX format



```
{
    "name": "mkringprogressview",
    "SPDXID": "SPDXRef-Package-swift-mkringprogressview-f1910c6e226e08f7",
    "versionInfo": "UNKNOWN",
    "supplier": "NOASSERTION",
    "downloadLocation": "NOASSERTION",
    "filesAnalyzed": false,
    "sourceInfo": "acquired package info from resolved Swift package manifest: /
    LoopWorkspace.xcworkspace/xcshareddata/swiftpm/Package.resolved",
    "licenseConcluded": "NOASSERTION",
    "licenseDeclared": "NOASSERTION",
    "copyrightText": "NOASSERTION",
    "externalRefs": [
        {
            "referenceCategory": "SECURITY",
            "referenceType": "cpe23Type",
            "referenceLocator": "cpe:2.
            3:a:mkringprogressview:mkringprogressview:*:*:*:*:*:*:*:*"
        },
        {
            "referenceCategory": "PACKAGE-MANAGER",
            "referenceType": "purl",
            "referenceLocator": "pkg:swift/github.com/maxkonovalov/
            MKRingProgressView.git/mkringprogressview"
        }
    ]
},
```

(b) SPDX format

Figure A.1: SBOM component information of Tidepool Loop application generated by Syft tool.

3. Sbomqs Evaluation Results

Table A.2 reports the evaluation results of sbomqs on CycloneDX and SPDX SBOMs.

Feature	CycloneDX Score	CycloneDX Desc	SPDX Score	SPDX Desc
comp_with_name	10.0/10.0	114/114 have names	10.0/10.0	107/107 have names
comp_with_supplier	0.0/10.0	0/114 have supplier names	0.1/10.0	1/107 have supplier names
comp_with_uniq_ids	10.0/10.0	All components have unique ID's	10.0/10.0	All components have unique ID's
comp_with_version	9.3/10.0	106/114 have versions	9.9/10.0	106/107 have versions
sbom_authors	10.0/10.0	doc has 1 author	10.0/10.0	doc has 2 authors
sbom_creation_timestamp	10.0/10.0	doc has creation timestamp	10.0/10.0	doc has creation timestamp
sbom_dependencies	0.0/10.0	0 dependencies listed	10.0/10.0	106 dependencies listed

TABLE A.2: Sbomqs Evaluation Results for CycloneDX and SPDX SBOMs