System & Software Security
Software Bills of Materials
Hands-On Project

Davide Baggio	s4426428
João Pereira	s4443322
Nuno Pereira	s4443330

November 2024

Contents

We confirm that this report was fully produced by the team members **Davide Baggio, João Pereira and Nuno Pereira** and we are jointly responsible for all content presented in this work. All used sources were attributed properly.

1 Introduction

Software Chain Security is an increasingly important challenge to tackle as early as possible in the Software Development process due to the catastrophic effect that vulnerabilities or security issues in downstream dependencies can cause to any software product. (Needs references) It is therefore crucial from a development standpoint to be aware of what components are included in any software project and the potential vulnerabilities that they might introduce. Many solutions have been developed to aid in that process, the prime example being *package-managers* [1], tools that help managing software dependencies and version conflicts between dependencies. Examples of such package managers are NPM (Node Package Manager) and Cargo [13, 14].

Another solution to dependency and vulnerability tracking are *Software Bills of Materials* (SBOMs) [2], detailed listings of dependencies, their relations, licensing information and other metadata pertinent to software products. These provide a standard format to work and process dependency information, allowing easier communication and shareability. Currently, 3 mainstream SBOM standards exist: OWASP's *CycloneDX*, Linux Foundation's *SPDX* and NIST's *SWID* [8, 9, 12].

In previous work [15], the authors performed a critical comparison on 5 articles from the literature [3–7] based on author-defined metrics. The authors provide insights on the state-of-the-art regarding SBOMs and provide guidelines for SBOM adoption and development, as well as future work that can be done in further research. Should add more sentences/explain this further

In an attempt to complement [15], this article reports on a hands-on comparison of the three mainstream SBOM standards by making use of available tools for each standard. SBOM generation tools for each standard were used on a set of N major Open-Source repositories found on GitHub and the resulting SBOM output files were compared, both between standards as well as between the tools of each standard.

2 Methodology

In this section, we detail the methodology used to perform the hands-on comparison between SBOM standards. In 2.1 we discuss the different available standards and the tools chosen to generate SBOMs for each one of them. Based on the tools picked we chose, in 2.2, the repositories on which the SBOMs will be generated.

2.1 Standards and tools

Each one of the three standard formats focuses on a specific part of the software supply chain, which can be reflected in the (meta)data that each standard stores and processes. The tools developed for each each standard also reflect these decisions. I felt that I had to fill this in with something

2.1.1 CycloneDX

CycloneDX [9] is an SBOM standard format developed by the CycloneDX Core Working Group and backed by the OWASP Foundation with a focus on "cyber-risk reduction" [9] and security [3]. The standard supports writing BOMs for several domains of software development, such as Software BOMs (SBOMs), Cryptographic BOMs (CBOMs),

Software-as-a-Service BOMs (SaaSBOMs), among others. Over 200 tools related to CycloneDX's SBOM format are available at CycloneDX's official tool webpage. Should this be a reference instead?

For this hands-on comparison, we limited our search to *Open-Source* tools as these are free to access and use. Out of 172 listed Open-Source tools, 2 were chosen: CycloneDX `cdxgen`[10] and `build-info-go` [11].

Other tools exists but they are either unrelated (SBOM analysis, VEX generation, ...), too specific (official SBOM generators for several existing programming languages and build tools) or too limited on the supported development environments.

2.1.2 SPDX

2.1.3 SWID

2.2 Repositories

To ensure a fair comparison between standards, we chose a representative set of major Open-Source repositories that could be analyzed by most, if not all, of the tools selected.

As such, ...

3 Conclusion

References

[1] Diomidis Spinellis. "Package Management Systems". In: *IEEE Software* 29.2 (2012). [Accessed 21-10-2024], pp. 84–86. DOI: 10.1109/MS.2012.38.

[2] Éamonn Ó Muirí. "Framing software component transparency: Establishing a common software bill of material (SBOM)". In: *NTIA, Nov* 12 (2019). [Accessed 15-10-2024].

[3] Boming Xia et al. "An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead". In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. [Accessed 15-10-2024]. 2023, pp. 2630–2642. DOI: 10.1109/ICSE48619.2023.00219.

[4] Nusrat Zahan et al. "Software Bills of Materials Are Required. Are We There Yet?" In: *IEEE Security & Privacy* 21.2 (2023). [Accessed 15-10-2024], pp. 82–88. DOI: 10.1109/MSEC.2023.3237100.

[5] Tingting Bi et al. "On the Way to SBOMs: Investigating Design Issues and Solutions in Practice". In: *ACM Trans. Softw. Eng. Methodol.* 33.6 (June 2024). [Accessed 15-10-2024]. ISSN: 1049-331X. DOI: 10.1145/3654442. URL: https://doi.org/10.1145/3654442.

[6] Berend Kloeg et al. "Charting the Path to SBOM Adoption: A Business Stakeholder-Centric Approach". In: *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*. ASIA CCS '24. [Accessed 15-10-2024]. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 1770–1783. ISBN: 9798400704826. DOI: 10.1145/3634737.3637659. URL: https://doi.org/10.1145/3634737.3637659.

[7] Trevor Stalnaker et al. "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. [Accessed 15-10-2024]. Lisbon, Portugal: Association for Computing Machinery, 2024. ISBN: 9798400702174. DOI: 10.1145/3597503.3623347. URL: https://doi.org/10.1145/3597503.3623347.

[8] National Institute of Standards and Technology (NIST). *Software Identification (SWID) Tagging — CSRC — CSRC — csrc.nist.gov*. https://csrc.nist.gov/projects/Software-Identification-SWID. [Accessed 15-10-2024].

[9] Open Worldwide Application Security Project (OWASP). *OWASP CycloneDX Software Bill of Materials (SBOM) Standard — cyclonedx.org*. https://cyclonedx.org/. [Accessed 15-10-2024].

[10] CycloneDX Core Team. *cdxgen documentation — cyclonedx.github.io*. [Accessed 24-11-2024]. URL: https://cyclonedx.github.io/cdxgen/.

[11] *JFrog Build Info - Build Info by JFrog — buildinfo.org*. [Accessed 24-11-2024]. URL: https://www.buildinfo.org/.

[12] Linux Foundation. *SPDX Linux Foundation Projects Site — spdx.dev*. https://spdx.dev/. [Accessed 15-10-2024].

[13] NodeJS Team. *NPM: Node Package Manager*. [Accessed 15-10-2024]. URL: https://www.npmjs.com/.

[14] Rust Team. *Cargo*. [Accessed 15-10-2024]. URL: https://doc.rust-lang.org/cargo/.

[15] Davide Baggio, João Pereira, and Nuno Pereira. "System & Software Security - Software Bills of Materials". [Accessed 24-11-2024]. N.D.