Editors: D. Balzarotti, davide.balzarotti@eurecom.fr | W. Enck, whenk@ncsu.edu | T. Holz, thorsten.holz@rub.de | A. Stavrou, astavrou@gmu.edu

Software Bills of Materials Are Required. Are We There Yet?

Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams | North Carolina State University

Executive order 14028 on improving the nation's cybersecurity highlights the software bill of materials (SBOM) as an essential security practice for software security. This article outlines the top five benefits and challenges of adopting SBOMs, identified by reviewing 200 Internet articles.

In December 2021, a zero-day remote code execution vulnerability (code-named Log4Shell) was identified in Apache Log4j, a popular logging library used by more than 35,000 Java packages.[1] Nearly every software organization immediately needed to discover which of its applications contained the affected version of the Log4j library so that it could mitigate its risk. Organizations with software bills of materials (SBOMs) identified the vulnerable component in a few hours, contrary to taking weeks to determine their risk.[2, 3] The Log4Shell vulnerability showed that SBOMs can be useful for keeping track of and mapping applications that depend on vulnerable dependencies and for shortening the time it takes to respond to the discovery of critical vulnerabilities.

SBOMs provide transparency and visibility to software components and dependencies. Executive order (EO) 14028 on improving the nation's cybersecurity states that "SBOMs are a formal record containing the details and supply chain relationships of various components used in building software." At the bare minimum, an SBOM includes the component name, publisher name, component version, other unique identifiers, dependency relationship, author of SBOM data, file name, license information, and time stamp. Open Web Application Security Project CycloneDX, Software Product Data Exchange (SPDX), and Software Identification Tagging (SWID) are three widely known SBOM standards. An SBOM standard is a schema designed to provide a common format for describing the composition of software in a way that is machine readable and consumable by other tools. SPDX was initiated in 2010, SWID in 2012, and CycloneDX in 2017.

Executive Order 14028 and SBOMs

In May 2021, the software industry witnessed a surge of practitioner interest in SBOMs, as EO 14028 identified SBOMs as one of the practices that enhance software supply chain security. The EO directed the Department of Commerce, in coordination with the National Telecommunications and Information Administration (NTIA), to publish the minimum elements for SBOMs. In July 2021, the NTIA released a report outlining the minimum elements, how an SBOM might assist in reducing risks, and options for future evolution. Subsequently, the National Institute for Standards and Technology (NIST) published the first version of the Secure Software Development Framework (SSDF), in February 2022, which lists the security practices a software organization can adopt, including the use of SBOMs.

On 14 September 2022, the Office of Management and Budget released a memo that required federal agencies to self-attest that software was built according to NIST SSDF guidance. Therefore, SBOMs are now required for software providers who conduct direct business with the U.S. federal government. However, SBOMs are likely equally important for those who do not sell directly to the U.S. government, as one of a company's clients might do business with a federal agency and use the product as a solution. Therefore, the ripple effect of the federal government's SBOMs requirements is likely to be significant.

Industry Consensus

While SBOMs have received notable policy attention from the U.S. government, software producers have yet to implement them fully. The Linux Foundation's SBOM readiness survey[4] in 2022 on 412 worldwide organizations showed gaps in familiarity with, production planning for, and consumption of SBOMs. For example, only 18% of organizations use SBOMs across nearly all their business segments and have standard practices that include using SBOMs.

Digital Object Identifier 10.1109/MSEC.2023.3237100
Date of current version: 15 March 2023

Other organizations use SBOMs to some extent, plan to adopt SBOMs, or have no plans to adopt SBOMs at all. Disconcertingly, 40% of the sample was unclear about whether the industry is committed to mandating SBOMs, raising concerns about whether practitioners are abiding by the EO governing SBOMs. Therefore, widespread SBOM adoption requires 1) identifying challenges associated with SBOM generation, 2) developing clear use cases, and 3) crafting guidelines and tooling to accommodate SBOMs use cases.

While SBOMs have received notable policy attention from the U.S. government, software producers have yet to implement them fully.

We conducted a gray literature (GL) review to investigate the challenges and benefits of the current SBOM adoption. The goal was to understand the benefits of SBOMs and why practitioners have not jumped into action to adopt them. We share the top five benefits and challenges extracted from the literature to aid federal and industry leaders in formulating action plans to address these issues and guide an SBOM transition for agencies and vendors.

Method

GL is defined as "literature that is not formally published in sources, such as books or journal articles."[5] We observe that practitioners often share their experience with SBOMs in blogs, videos, white papers, and webpages as GL, but no systematic peer-reviewed SBOM research studies have been published.

We conducted a GL review in which two authors independently reviewed Internet articles, blogs, and interviews to extract the challenges and benefits of SBOM adoption. Our search technique had two query strings: 1) "challenges to adopt SBOM" and 2) "benefits to adopt SBOM." We selected the top 100 results determined by the Google search engine's page rank algorithm as a search stopping criterion. To avoid bias from an author's browsing history, we browsed in Google Chrome's incognito mode. In total, we collected 200 Internet articles, 100 for each query string, on 25 November 2022. We used the following inclusion criteria to collect relevant Internet artifacts for this study:

- The article is written in English.
- The article content is relevant to SBOM adoption.
- The article discusses at least one benefit or one challenge of SBOM adoption.
- The article is not a duplicate of another artifact.
- The article is not a product of a vendor advertisement.

Using our inclusion criteria, we identified 62 articles[6] and applied open coding techniques. Two authors individually reviewed 62 articles and recorded the challenges and benefits of SBOM adoption independently. Then, two reviewers cross-checked the identified challenges and benefits from each article and resolved any disagreements. Finally, the first author grouped the identified challenges and benefits into five categories.

Benefits

SBOMs provide transparency and visibility and help practitioners understand what pieces of software are being used, the relationship among them, and the software's exposure to security risks. We now provide information about the top five benefits of SBOMs, as highlighted in 50 of the 62 Internet articles. We provide in parentheses the number of articles that mention each benefit.

Benefit 1: Dependency Management (39)

SBOMs provide explicit identification of external dependencies, which is useful for tracking open source, commercial, and custom-built software dependencies across applications. Transparency in a dependency tracking system helps practitioners manage products more efficiently, identify security risks for quick remediation of direct and transitive dependencies, and ensure that developers use approved code and sources. Using SBOMs to share infrastructure and data may also save time by making it easier for departments to work together and facilitate code reuse.

Benefit 2: Vulnerability Management (29)

With SBOMs, teams can identify and prioritize specific vulnerabilities in potentially risky dependencies. Security experts can use SBOM artifacts to determine the impact on code when a critical vulnerability is disclosed. Therefore, incident response personnel can identify vulnerabilities and prioritize mitigation, e.g., as with the Log4j vulnerability.[1] SBOMs also allow leaders to respond quickly and confidently when dealing with board members, customers, investors, and regulators.

The Vulnerability Exploitability Exchange (VEX)[7] is a machine-readable companion artifact to SBOMs. VEX was developed by the NTIA and the Cybersecurity and Infrastructure Security Agency to help suppliers determine whether a specific product is impacted by a specific vulnerability, with an emphasis on the vulnerability's exploitability. With VEX and SBOMs, organizations can identify vulnerabilities that pose an immediate risk, enabling a deprioritization of vulnerabilities that are not exploitable.

Benefit 3: Risk Management (21)

SBOMs can be used as a risk management tool to help detect, recognize, identify, respond to, and remedy threats in supply chains. The software industry is increasingly promoting adopting strategic and proactive risk management rather than the traditional reactive approach. Because of the expansion of the digital attack surface, companies and agencies need to rely more on informed risk management. SBOMs help analyze, evaluate, control, and find security risks in software early on in the production cycle.

Practitioners can have a more informed picture of the risk factors a specific BOM brings to an environment. Moreover, the risk factor can vary based on whether or not the SBOM is modified, given that new vulnerabilities are discovered regularly. By offering transparency, SBOMs can facilitate the detection of these newly identified threats and the evaluation of the magnitude of the breach. For instance, if a new vulnerability is entered into the National Vulnerability Database, SBOMs provide a way for buyers, vendors, and software users to track software dependencies throughout supply chains, identify the vulnerability location, and anticipate emerging risks.

Benefit 4: License Management (13)

Legal teams often need the security team to have clear visibility to ensure that no direct and transitive dependencies are using licenses that break the organization's legal policies. SBOMs aid compliance teams in responding to license claims and audits.

Benefit 5: Competitive Advantage (6)

SBOM transparency aids in making informed purchasing decisions. Since SBOMs are now required for organizations doing business with federal agencies, industry leaders will likely request an SBOM from vendors. In this case, software vendors could gain a competitive advantage through the production of an SBOM. Since a comprehensive SBOM will make it easier to identify risky dependencies in the supply chain, companies that produce more secure components will be able to charge premium prices. Software vendors may be compelled to reengineer their products from a security perspective to maintain their competitive edge.

Since SBOMs are now required for organizations doing business with federal agencies, industry leaders will likely request an SBOM from vendors.

Challenges

All authors of the 62 articles mentioned that SBOMs are critical for enhancing software security, particularly in light of the EO's call for software producers to produce them. In this section, we discuss the challenges with SBOMs as conveyed by 41 Internet articles. We provide in parentheses the number of articles that mention each challenge.

Challenge 1: SBOM Tools (29)

Many practitioners commented that the industry lacks the tools to accurately and automatically generate SBOM data at scale for a product's supply chain. The SPDX and CycloneDX standards have around 200 tools, most of which are efficient in generating SBOMs for a single component. Hence, the tooling issue is largely associated with a need for automation at scale, accuracy, and runtime BOMs.

SBOM article authors indicated that most of these SBOM tools have a high false positive rate. For example, many tools' approaches to creating an SBOM rely on guessing and using heuristics and information from the top-level component to generate an "ingredient" list. The SBOM tools often rely on discovering dependencies from package managers [e.g., Pip and Node Package Manager (NPM)], which have different standards. The package configuration files may not document all the dependencies used by the software, let alone transitive dependencies. Another example is software vendors generating an SBOM by scanning compiled artifacts: they may miss statically linked binaries that do not include dependency metadata. An SBOM can provide a false sense of "completeness." Therefore, given the poor state of the original metadata from ecosystems and compiled artifacts, practitioners indicated that SBOM tools often reflect a "garbage in, garbage out" status.

The lack of automation to detect modifications makes SBOM adoption challenging. An SBOM needs to be updated whenever a change is made to a component, including code updates, vulnerability patches, new features, and other modifications. Unfortunately, such activities are typically done manually, which is a highly difficult task given that changes can happen at any time. Organizations can benefit from having a "runtime BOM" containing a list of outbound connections, a list of files touched, the presence of ports listening in, and so on. These metrics are important for security teams to know and track because they increase the attack surface.

SBOM tooling does not support different data requirements and account for flexibility among different use cases. Scaling SBOM generation is difficult for vendors if different clients require different information. For instance, for an organization that cares deeply about software licenses, vendors need to deliver accurate licenses. For an organization that cares about vulnerabilities, vendors need a cost-effective way to deliver less manual VEX data. On top of that, vendors lack tools for ingesting SBOM and VEX data. Therefore, developers are forced to spend more time on manual tasks. Overall, our findings indicate that practitioners want the industry to prioritize the production of tools that generate accurate SBOMs automatically for different use cases.

Practitioners want the industry to prioritize the production of tools that generate accurate SBOMs automatically for different use cases.

Challenge 2: Lack of SBOM Interoperability (13)

According to practitioners, a major impediment to adopting SBOMs is the need for standards in data quality, data exchange formats, and tooling. While SBOMs have garnered notable policy attention from the government, the attention is primarily focused on what an SBOM should contain; discussions around data exchange standardization, workflow automation, implementation, and processes that are often left out.

The difficulties in mapping software components for different versions and ecosystems due to a lack of standards in a package's unique identifier generation were the second-most frequently stated challenges in our study. As of 2022, the industry had three prominent SBOM standards—SPDX, SWID, and CycloneX—but these standards do not constitute any agreement on how to generate unique SBOM identifiers for packages and versions. For example, a standard might state that the SBOM for each package should contain Common Platform Enumeration or a package uniform resource locator (URL) or both. According to these standards, two SBOM formats can use different unique identifiers for the same component, and two SBOMs using different components can have the same identifier because the industry lacks standard naming conventions across ecosystems and SBOM formats. For instance, practitioners could publish a Rust package to Cargo and a JavaScript package to NPM using the same name: "Package-AA & version X.Y.Z." Then, the question becomes how to differentiate the SBOMs of these two packages in terms of unique identifiers across different standards.

Moreover, SBOM standards lack uniformity in the display and mapping of identical data across tools. Sometimes SBOM tools use the package URL (e.g., Cargo and NPM package URL) and repository URL as an SBOM identifier for the package version, and others do not. Many tools do not provide these URLs because they are not required by the minimum data elements published by the NTIA. Another observation is that data can be located in different fields. For example, sometimes the package URL can be used in a field defined for the package URL, sometimes as the name, and even in a totally different data field and location. Without a standard mapping between data fields and data, SBOMs generated by different tools for the same package may appear distinct despite containing the same data.

The lack of a uniform and automated platform for SBOM sharing that stores unique identifiers for different files and suppliers is another challenge. If each tool and vendor does not offer more than one SBOM format, implementing SBOMs in the software development lifecycle can be challenging. For example, if software producers obtain different SBOM formats from upstream software vendors, retrieving the different SBOM formats and reconciling the software components can be difficult. Practitioners need a standard method so that vendors can follow the same system and data format and rely on its accuracy.

Therefore, data quality, global mapping, and the lack of a shared platform in the SBOM ecosystem are crucial barriers preventing the industry from adopting and consuming SBOMs. Going forward, emphasis should be placed on achieving the interoperability of SBOMs across different formats, vendors, and tools.

Challenge 3: Value of SBOMs (14)

Practitioners expressed a lack of data to back up the effectiveness and value of SBOMs, preventing practitioners from adopting SBOMs. At the time of the study, SBOMs were a requirement for software vendors, but there were no regulatory requirements for SBOMs purchasers on what to do with the information provided by an SBOM. Even though this information is useful if an organization does not have a process to consume it, having an SBOM provides minimal value.

A basic SBOM can be used as a building block to determine whether a software product is affected by identified vulnerabilities. An SBOM alone does not list vulnerabilities. Instead, the information in the SBOM must be cross-referenced with information in vulnerability databases. Existing software composition analysis tools have started to perform these checks; however, they tend to be noisy when a vulnerable dependency is not used in a vulnerable way. VEX information can help bridge this gap, but existing VEX generation and analysis are manual.

Furthermore, SBOMs can aid in identifying legacy software and abandoned and unmaintained dependencies, but the concern is what to do with these dependencies. According to multiple studies and reports,[8, 9, 10] over 70% of open source software packages are unmaintained, abandoned, and never updated. Should practitioners begin maintaining such a large number of unmaintained open source software packages, which is resource intensive, or begin developing new components, which is time intensive, or take no action?

Challenge 4: Time Consumption (7)

The difficulties in tracking components and dependencies and the lack of standard tools and automation make SBOM generation time-consuming. SBOM standards and tool implementations are based on existing practices, and today, these practices and the associated tooling are inconsistent. As a result, software development teams need to put in a lot of effort to generate accurate SBOMs. For example, software producers often have to manually generate an SBOM by listing components in the product's supply chain because end products do not contain an accurate mapping of direct and transitive dependencies.

Furthermore, maintaining code bases with dozens and even hundreds of dependencies and third-party components is a tiresome and time-consuming task. Historically, many developers often included third-party components in an application, without documenting the change. As a result, current developers may not be familiar with the entire code base. Managing and tracking this large amount of SBOM information (scaling) places a large burden on developers because the software industry does not have any standard centralized repository to store SBOMs across product teams and applications.

Another factor that prolongs the SBOM generation process is a lack of training and resources to generate and maintain SBOMs. For example, small software vendors selling to the federal government and volunteer open source software developers could be affected differently by any forthcoming SBOM requirements, due to a lack of resources and incentives. Practitioners mentioned that organizations also struggle with determining who is ultimately responsible for open source and third-party component management.

Most tools produce an SBOM as a static list of dependencies in an application or container image. Since SBOMs are not directly referenced by vulnerability databases, making vulnerabilities transparent in an SBOM is difficult and time-consuming.

Without a standard mapping between data fields and data, SBOMs generated by different tools for the same package may appear distinct despite containing the same data.

Challenge 5: Risks of Transparency (4)

Practitioners are concerned that SBOMs may expose some organizations to new risks by revealing internal information to competitors and bad actors. In addition to the general hesitation of any developer or organization to expose vulnerabilities and outdated unpatched dependencies to clients, practitioners mentioned the following tradeoffs of sharing SBOMs:

- Since SBOMs effectively provide a road map to the architecture and components of an application, vendors may be forced to deal with disputes over trade secrets and reverse engineering involving SBOM information.
- There are concerns regarding how software purchasers will protect the information SBOMs provide and preserve vendors' intellectual property rights.
- Will the risk of distributing "incorrect" SBOMs, due either to negligent practices or understandable errors, result in penalties?

If the software industry wants vendors to take on this cost, vendors require assurance to protect their intellectual property and help reduce the generation and support costs of SBOMs.

SBOMs are a step toward transparency and vigilance for software developers and users to identify and address risks. SBOM services are increasingly used in vulnerability management and risk management for third parties. The federal government is a large consumer of software, and its expectation of SBOM disclosure from suppliers will push SBOM adoption among many vendors and purchasers.

However, as of 2022, SBOM standards were inadequate for large-scale adoption. The future of SBOM usage, disclosure, sharing, and consumption is largely dependent on industry standards and practices. While the problems SBOMs could solve are compelling, without addressing the challenges associated with each benefit, industry-wide SBOM adoption is difficult. Given the intertwined challenges of accuracy, timeliness, complexity, motivation, and values, software producers need strong guidelines and standardization efforts to ensure consistent data requirements to meet the use cases SBOMs intend to serve. As an industry, practitioners should partner with tooling providers and policy makers to focus on solving the top SBOM use cases and challenges. Practitioners can do this by standardizing the requirements for each use case and influencing guidelines and tooling to accommodate the requirements. Interoperability, vendor cross compatibility, and automation are key to success if the industry wants SBOMs at scale. ■

References

1. *Apache Log4j Security Vulnerabilities*. (2021). Log4J. [Online]. Available: https://logging.apache.org/log4j/2.x/security.html
2. P. Roberts. "Log4j is why you need a software bill of materials (SBOM)." Reversing Labs. Accessed: Feb. 6, 2023. [Online]. Available: https://www.reversinglabs.com/blog/log4j-is-why-you-need-an-sbom
3. L. Vaas. *One Year After Log4shell, Firms Still Struggle to Hunt Down Log4j*. (2022). Contrast Security. [Online]. Available: https://www.contrastsecurity.com/security-influencers/one-year-after-log4shell-firms-still-struggle-to-hunt-down-log4j
4. *The State of Software Bill of Materials (SBOM) and Cybersecurity Readiness*. (2022). Linux Foundation. [Online]. Available: https://www.linuxfoundation.org/research/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness
5. V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 106, pp. 101–121, Feb. 2019, doi: 10.1016/j.infsof.2018.09.006.
6. E. Lin, 2023, "List of SBOM articles reviewed for grey literature, doi: 10.5281/zenodo.7513103.
7. "Vulnerability-exploitability exchange (vex)," Nat. Telecommun. Inf. Admin., Washington, DC, USA, 2021. [Online]. Available: https://www.ntia.gov/files/ntia/publications/vex_onepage_summary.pdf
8. *State of Software Security v11: Open Source Edition*. (2022). Veracode. [Online]. Available: https://info.veracode.com/fy22-state-of-software-security-v11-open-source-edition.html
9. N. Zahan et al., "What are weak links in the NPM supply chain?" in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng., Softw. Eng. Pract. (ICSE-SEIP)*, 2022, pp. 331–340, doi: 10.1109/ICSE-SEIP55303.2022.9794068.
10. N. Zahan et al., "Preprint: OPENSSF scorecard: On the path toward ecosystem-wide automated security metrics." 2022, *arXiv:2208.03412*.

Nusrat Zahan is a Ph.D. student in the Department of Computer Science at North Carolina State University, Raleigh, NC 27695 USA. Her research interests include software security. Zahan received a B.Sc. in electronics and communication engineering from Khulna University of Engineering & Technology (KUET), Bangladesh. She is a member of ACM. Contact her at nzahan@ncsu.edu.

Elizabeth Lin is a Ph.D. student in the Department of Computer Science at North Carolina State University, Raleigh, NC 27695 USA. Her research interest is software supply chain security. Lin received a B.Sc. in computer science from National Chengchi University, Taiwan. Contact her at etlin@ncsu.edu.

Mahzabin Tamanna is a Ph.D. student in the Department of Computer Science at North Carolina State University, Raleigh, NC 27695 USA. Her research focuses on investigating security issues in software supply chains. Tamanna received an M.Sc. in computer science from North Carolina Agricultural and Technical State University. Contact her at mtamann@ncsu.edu.

William Enck is a professor in the Department of Computer Science and codirector of the Secure Computing Institute at North Carolina State University, Raleigh, NC 27695 USA. His research interests include system security. Enck received a Ph.D. in computer science and engineering from Penn State University. Enck is a member of the USENIX Association and ACM and a Senior Member of IEEE and ACM. Contact him at whenck@ncsu.edu.

Laurie Williams is a distinguished university professor and codirector of the Secure Computing Institute, North Carolina State University, Raleigh, NC 27695 USA. Her research interests include software security. Williams received a Ph.D. in computer science from the University of Utah. She is a Fellow of IEEE and ACM. Contact her at lawilli3@ncsu.edu.