THE CYBERSECURITY BENEFITS OF LEVERAGING A SOFTWARE BILL OF MATERIALS

Drew Buttner and Bob Martin

Use of a Software Bill Of Material (SBOM) can reduce financial, personnel, and reputational risks incurred by using unknown software. It enables system engineering, acquisition, and cybersecurity teams to better understand the make-up of their critical infrastructure and to automate tasks to help assess and determine associated risk. Finally, an SBOM can be a starting point to map to other sources of information that we might care about. For example understanding the political jurisdictions of developers or looking for single-contributor Open Source Software libraries.

This paper focuses on the benefits of adopting and using an SBOM to increase software transparency, resulting in increased software component trustworthiness and overall cybersecurity.

INTRODUCTION

Today's systems are composed of complex and interdependent software, where thousands of components from hundreds of different vendors are brought together to achieve development objectives. Lack of systemic visibility into the provenance, composition, and functionality of software increases organizational risk associated with its use and drives up the costs of protecting the organization. [1] Unknown components may cause systems to perform in unexpected ways and can create prolonged exposure to attack when the components have publicly known vulnerabilities. The recent Log4Shell incident brought new light to the challenges we face due to endemic vulnerabilities. [2]

Knowledge of components and their origin and history allows for diverse organizational risk assessment [3] and mitigation of issues while supporting software providers as they work to address problems. In our increasingly interconnected world, risk and associated cost due to a lack of transparency impact not only individuals and organizations but also collective goods like public safety and national security.

The foundation for achieving software transparency comes from securely capturing, with cryptographic assurance, details from across the software lifecycle. This includes data about the pedigree and provenance of individual components, the respective source of components, and their chain of custody across the software lifecycle.

AN SBOM IS A FORMAL, MACHINE-READABLE HIERARCHICAL INVENTORY OF SOFTWARE COMPONENTS AND INFORMATION ABOUT THOSE COMPONENTS.

Simply capturing more metadata is helpful, but effectively using this data requires both automated consumption and appropriate policy enforcement. This requires not just machine readability but also agreed upon semantic interpretation of the data, which further requires defining data specifications and standardization.

At the heart of this process is the concept of an SBOM to enable risk and cost reduction by: [1]

- Improving the ability to identify vulnerable components that contribute to security incidents.
- Reducing unplanned and unproductive work due to convoluted supply chains that bring software provenance and integrity into question.
- Facilitating the identification of out-of-date, single-maintainer or unsupported components.

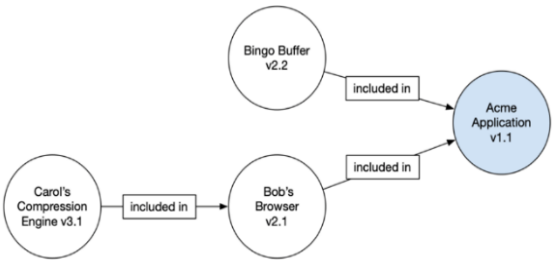
SOFTWARE BILL OF MATERIALS

A Bill of Materials (BOM) is a nested inventory of components that make up items such as software, hardware, or systems. A standard BOM is meant to be a formal record containing the details and supply chain relationships of various components used in building an item.

The software version of a BOM (referred to as an SBOM) provides those who produce, purchase, and operate software with information that enhances their understanding of the software components and associated supply chain.

Reports written by the Department of Commerce's National Telecommunications and Information Administration (NTIA) [1] [4] provide a definition of an SBOM:

> An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships. These inventories should be comprehensive – or should explicitly state when they cannot be. SBOMs may include open source or proprietary software and can be widely available or access restricted.



> SBOMs should also include baseline attributes with the ability to uniquely identify individual components in a standard data format. The most efficient generation of SBOMs is as a byproduct of a modern development process. For older software, semi-automated methods exist.

This enhanced understanding of the supply chain enables multiple benefits, most notably the potential to track known and newly discovered vulnerabilities and risks. SBOMs will not solve all transparency problems, but they will form a foundational data layer on which further tools, practices, and assurances can be built.

The Department of Commerce prepared a report on "The Minimum Elements for a Software Bill of Materials (SBOM)" [6] as part of its response to the United States Presidential Executive Order 14028 on Improving the Nation's Cybersecurity. [5] The minimal elements defined in the report are the starting point for a viable automatable SBOM format that the Executive Order requires software vendors to begin providing. The report also discusses "Automation Support" and lays out the types of issues and opportunities the market is addressing to bring automation to this pressing problem area. Finally, the report covers "Beyond Minimum Elements: Enabling Broader SBOM Use Cases" and lays out the larger possibilities for integrity that cryptographic hashes can provide, as well as other types of information about pedigree, provenance, licensing, and vulnerabilities that an SBOM can convey as it travels through the software supply chain.

CYBERSECURITY BENEFITS

SBOMs facilitate a deeper understanding of software and enables achievement of the following cybersecurity benefits.

1. Safeguard our critical systems, vehicles, aircraft, spacecraft, vessels, and infrastructure: Cybersecurity teams know exactly what software is included and where the software originated. This understanding allows teams to design targeted tests and to track potential threats against the

software. Without this understanding, software is an unknown black-box that may perform in unexpected and undesired (i.e., malicious) ways.

2. Rapidly enact protective measures of deployed vulnerable software: Software is never perfect, and vulnerabilities are discovered on a daily basis. Adversaries leverage these vulnerabilities to achieve their malicious goals. SBOMs provide the information that drives automated checking for known vulnerabilities within the software. This knowledge provides cybersecurity teams the ability to enact protective measures and stop adversarial attacks.
3. Rapidly upgrade and patch software: Out-of-date and unpatched software is a prime target of adversaries. Cybersecurity teams understand this threat and try to mitigate it by upgrading and patching software as quickly as possible. SBOMs allow cybersecurity teams to know the current version of software (and its embedded components) and plan for and automate its upgrade.
4. Ensure acceptable software development practices: Assessment of software can be a long and difficult task. In support of this assessment, cybersecurity teams often look at the development practices used by the creators of the software. SBOMs provide an understanding of all the software included in a system, and thus enable cybersecurity teams to search out relevant information about development practices.
5. Ascertain authenticity and risk of the software: The provenance of software is important in the understanding of risk and verifying the integrity of acquired software is a critical component in an assertion of provenance. Supply chains that involve many intermediate steps complicate this matter. SBOMs provide a way to pass cryptographic proofs of integrity for all the software included in a system, thus providing assurances of the declared provenance.

DESIRED INFORMATION

To support the previously stated cybersecurity objectives, an SBOM must contain specific information. The minimum elements specified in NTIA's report covers most of this necessary information. These are:

- Supplier Name – The entity that creates, defines, and identifies components.
- Component Name – Designation assigned to a unit of software defined by the original supplier.
- Version of the Component – Identifier used by the supplier to specify a change in software from a previously identified version.
- Other Unique Identifiers – Other identifiers that are used to identify a component or serve as a look-up key for relevant databases. Examples of commonly used unique identifiers are Common Platform Enumeration (CPE), Software Identification (SWID) tags, Git Object ID (gitoid)/GitBOM, and Package Uniform Resource Locators (PURL).
- Dependency Relationship – Characterizing the relationship that an upstream component X is included in software Y.

- Author of SBOM Data – The name of the entity that creates the SBOM data for this component.
- Timestamp – Record of the date and time of the SBOM data assembly.

In addition to these minimum elements, the following additional elements beyond those identified in the Department of Commerce's report are helpful in performing advanced assessment, specifically in support of the use cases assessing development practices and software provenance:

- Source Location – A URL for the source code repository that includes a history of code submissions. (e.g., a GitHub URL)
- Cryptographic Signature – A signature used to verify the integrity of the delivered software.

Finally, there is the question of how "deep" or "complete" the SBOM component information is or needs to be. The supplier has first hand knowledge of what they incorporated into their product but may not have details about what is within those components. Flowing the requirements for SBOM information to a supplier's supplier can provide that next layer of completeness about what is in the product being operated but what about the next step below that? In a perfect world one wants to know the components all the way down, but in practice getting down to at least the level of open source components may address the vast majority of transpaency issues. With that information and investigation into the open source component tree can be performed on a repeating schedule leveraging automated processes.

EXISTING FORMATS

As of April 2022 there are two primary data formats being used to generate and consume SBOMs. These are:

a) Software Package Data eXchange (SPDX) is an open standard for communicating SBOM information, including com-ponents, licenses, copyrights, and security references.

b) CycloneDX is a lightweight SBOM open standard designed for use in application security contexts and supply chain component analysis.

Both of these formats are open and available, support all of the desired minimum elements, have tools that support SBOM generation and use, and continue to evolve and improve.

Both SPDX and CycloneDX efforts have active communities supporting the development of the specifications and the tooling to implement each.

SPDX is currently publishing a 2.2.2 release to fix some issues, and is finishing a 2.3.0 release to start implementing capabilities to meet Executive Order 14028's stated needs regarding linkage to vulnerability information. These non-breaking initial capabilities being included in 2.x releases are the on-ramp to more capable future implementations of SPDX.

The Linux Foundation is working to create this new version of SPDX (tentatively referred to as version 3.0) that will be an architectural change from the previous 2.x approach. The goal of the redrafted version of SPDX is to become a core BOM capability that has discrete profiles of capabilities and supporting information layered on top of that core. This approach will allow SPDX to be used for software, hardware, system, and network bills of

materials and addresses any emerging needs in industry and government for transparency about systems in a form that is machine processable.

The SPDX 3.0 effort is run by a steering committee that lightly oversees three top-level teams: Technical, Legal, and Outreach. The Technical Team is focused on five information profiles with working groups established to address each: Core, Licensing, Defects/Vulnerabilities, Usage, and Linkage. These working groups meet weekly for a couple of hours depending on the pace of issues being addressed.

Every month, there is a general SPDX meeting where an invited speaker presents something of interest to the whole group and then each of the top-level teams reports out on progress and issues.

The CycloneDX process is similar to that of a Change Management Process. Community ideas result in the creation of a Request For Change (RFC), which is discussed, adjudicated, and eventually voted on. Accepted proposals are then implemented and reviewed resulting in a revised specification. The bulk of the CycloneDX organized teams are comprised of developers/contributors, but an industry working group is also part of the CycloneDX process. No published schedules or timelines are available, but a public mailing list and Slack channel are used for discussions.

There are active tasks within both SPDX and CycloneDX to provide for loss-less transformation between the two formats. If this can be achieved and maintained going forward, the choice between the two formats will be inconsequential and left up to the user. However, if inter-translation is lost for key SBOM capabilities, then the tool market will decide which approach delivers the needed functionality – both to those developing and maintaining software and to those receiving any utilizing software in their products, services, and organizations.

The SPDX and CycloneDX communities have many overlapping members, which reinforces the desire for inter-exchangeability.

PREPARING FOR THE FUTURE

Starting from scratch and establishing an SBOM culture will not be easy, but the benefits to an organization will outweigh the initial cost. As with most efforts, it must begin with business or mission objectives and the desire to meet them. These objectives and the acceptance of the end goal can move and organization forward with an expectation for increasing compliance. To realize the cybersecurity benefits afforded by an SBOM, we recommend that system engineering, acquisition, and cybersecurity teams push toward these objectives.

1. **Require vendors to provide SBOMs with their software.** A software vendor has the most accurate and available information regarding the makeup of a given software product. A vendor can most efficiently produce a correct and complete SBOM. Build tools will eventually support vendors' development teams and automatically create SBOMs based on the actual build of the software. The vendor can also use the SBOM to provide the cryptographic information necessary to prove software integrity. Acquisition and engineering teams must look to only acquire and leverage software that comes with an associated SBOM. In parallel, engineering teams must verify the claims within SBOMs provided by vendors. Software

Composition Analysis tools are positioned to help with this.

2. **Create SBOMs as needed.** Unfortunately, SBOMs are not available today from all vendors. To move the bigger effort forward, SBOMs can be created by system engineering or cybersecurity teams as a short-term solution. Software Composition Analysis tools should be leveraged, as manual creation of SBOMs is a lot of work and is not sustainable in the long run.
3. **Leverage assessment tools that use SBOMs as input.** Cybersecurity teams tasked with assessing software leveraged within a system must look to use tools that take advantage of the information contained within an SBOM. This enables assessment on all components within the software and allows assessment beyond a single top-level vendor.
4. **Assess software beyond the source code.** Source code analysis is a beneficial and encouraged activity. However, due to well-understood limitations and heavy resource requirements, it often cannot be performed at the desired level. To compensate, cybersecurity teams must look beyond the source code and factor development practices and project status (e.g., is there active development?) into an overall understanding of risk.
5. **Integrate SBOMs into continuous monitoring activities.** Assessment should not be a one-time activity. New vulnerabilities are discovered daily, and new versions of software are released in response. SBOMs enable cybersecurity teams to perform assessment of the complete software picture on a continuous basis, and allow system engineering teams to be alerted about relevant upgrades.

With the adoption of standard-based SBOMs across the software development and cybersecurity communities, new challenges will emerge. Data integration, management, conformance, and sharing will become issues that need to be addressed to truly realize the benefits that SBOMs bring to cybersecurity. Other areas of cybersecurity will solve these challenges; monitoring and leveraging those solutions may give an organization a head start in meeting the SBOM demands of the future.

Finally, conversations must be started with cloud-based capability providers regarding what they can offer regarding software transparency, an-prem software becomes a smaller part of all of an organization's software-enabled capabilities in the future.

CONCLUSION

Integrating SBOMs into cybersecurity tasks enables the automated identification of vulnerable components that contribute to security incidents, reduces unplanned and unproductive work due to convoluted supply chains that bring provenance and integrity into question, and facilitates the identification of out-of-date components. Automation within these activities saves money as activities are completed in less time and with fewer errors. Future attacks against our systems are stopped before they even start, as components are updated and patched before the adversary can take advantage of known issues.

SBOMs enable system engineering, acquisition, and cybersecurity teams to better understand the software that makes up our critical infrastructure and to automate tasks to help assess and determine associated

risk. The community is working to standardize the SBOM format and the minimum information that is provided within, and to gain acceptance from commercial vendors and tool providers.

By preparing for an SBOM-enabled future, organizations can be ready to take advantage of the benefits and the cost savings that go along with them.

REFERENCES

[1] Framing Working Group of the NTIA Software Component Transparency multistakeholder process, "Framing Software Component Transparency: Establishing a Common Software Bill of Material (SBOM)," 12 November 2019. [Online]. Available: https://www.ntia.gov/files/ntia/publications/framingsbom_20191112.pdf.

[2] D. Wilburn and C. Schmidt, "Log4Shell and Endemic Vulnerabilities in Open Source Libraries," March 2022. [Online]. Available: https://www.mitre.org/sites/default/files/publications/pr-22-0797-log4shell-and-endemic-vulnerabilities-in-open-source-libraries.pdf.

[3] D. Geer, J. S. Meyers, J. Kazil, T. Pike, “Should Uncle Sam Worry About ‘Foreign’ Open-Source Software? Geographic Known Unknowns and Open-Source Software Security,” August 2022, [Online]. Available: https://www.lawfareblog.com/should-uncle-sam-worry-about-foreign-open-source-software-geographic-known-unknowns-and-open-source.

[4] Awareness and Adoption Working Group of the NTIA Software Component Transparency multistakeholder process, "SBOM at a Glance," 27 April 2021. [Online]. Available: https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_apr2021.pdf.

[5] "Executive Order on Improving the Nation’s Cybersecurity," 12 May 2021. [Online]. Available: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

[6] The Department of Commerce, "The Minimum Elements For a Software Bill of Materials (SBOM)," 12 July 2021. [Online]. Available: https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf.

ACKNOWLEDGEMENTS

The authors extend thanks to their MITRE, government, and industry colleagues for their advice, review, and expertise.

About the Authors

Drew Buttner *leads the software assurance technical capability area within MITRE. Mr. Buttner created the internal secure code review program at MITRE, and draws on over 20 years of experience across static code analysis, software security weaknesses, and secure coding practices.*

Robert Martin *leads software supply chain security efforts within MITRE and with industry and is the elected chair of the Industry IoT Consortium Steering Committee. Mr. Martin created the community standard for software security weaknesses used globally as well as over 50 global standards addressing the interplay of enterprise risk management, cybersecurity, and critical infrastructure protection.*

About MITRE

MITRE's mission-driven teams are dedicated to solving problems for a safer world. Through our public-private partnerships and federally funded R&D centers, we work across government and in partnership with industry to tackle challenges to the safety, stability, and well-being of our nation.

The views, opinions, and/or findings contained herein are those of the author(s) and should not be construed as an official government position, policy, or decision unless designated by other documentation.