Minimum Requirements for SBOMs

Authors:
Eric Brewer, Google Fellow and VP Infrastructure, ebrewer@google.com
Dan Lorenc, Staff Software Engineer, dlorenc@google.com
June 17, 2021 in response to the NTIA request for comments

Overview

The efficient generation and consumption of Software-Bill-of-Materials (SBOM) documents requires a clear understanding of the intended use-cases of the data by the consumer. The benefits of an SBOM differ greatly depending on the type of software being produced (Open Source vs. Proprietary), and how it is consumed (Box Software vs. Software-as-a-Service). This paper outlines the minimum requirements for data included in an SBOM, broken down by these software categories and use-cases. In addition to defining the minimum characteristics, we also believe guidance needs to be given on the *maximum* amount and types of data to include to allow for efficient consumption, and suggest some additional considerations around how the data should be produced to ensure accurate, reliable and actionable information.

Use Cases

To start defining the minimum requirements for an SBOM, we have to first understand exactly who will use the data, how they will use it, and for what. The Roles and Benefits for SBOM Across the Supply Chain (2019) published by the NTIA contains an exhaustive list of the potential benefits of SBOMs by various actors across the supply-chain. We believe that document contains two particularly clear use-cases that effectively address critical supply-chain security issues facing our industry, which are not addressed by existing mechanisms. **Focusing on these two use-cases will help the software community and SBOM tooling vendors direct their efforts toward quickly producing SBOMs that contain the most useful data for consumers**. These use-cases are: *Vulnerability Lifecycle Management* and *More Informed Purchasing*.

Vulnerability Lifecycle Management empowers a software end-user to become proactive in responding to security events. Without an understanding of what components are in the software they use (including exact versions and metadata), consumers cannot take remediation actions. The NTIA paper showed this in the "Time To Remediation" case study, as well as how the benefits of Vulnerability Lifecycle Management can compound as supply chains get longer.

More Informed Purchasing allows a customer to get an understanding of the high-level architecture and security posture of software components they are evaluating, and to use this as a factor in decision making. Fine-grained data is not likely to be as useful here, but aggregate information such as the average age of dependencies, the mean time to remediate security events, or the use of modern tooling like memory-safe programming languages can be useful when choosing among vendors.

Box vs. SaaS

If we examine the core use cases outlined above, the data required in an SBOM becomes easier to reason about. There is another dimension to consider though: the method of software distribution. The division of roles and responsibilities between the software vendor and the purchaser, particularly around responding to security vulnerabilities, vary greatly depending on the type of software purchased. For example, the vendor is typically responsible for the entire security patch lifecycle and vulnerability management for centrally-managed SAAS. The consumer needs to be aware of potential issues and the timeline for resolution, but the vendor owns the process from initial report to deployment of the fix.

At the opposite end of the spectrum, traditional "Box" (self-hosted) software places much more of a burden on the end user. Vendors may respond to alerts and release fixes, but these fixes must be monitored, ingested, evaluated, and eventually distributed *by the consumer*. In this model, automation through accurate supply-chain metadata such as SBOMs is critical to increasing the speed at which organizations can respond to security events.

The following two tables depict the roles and responsibilities for each party when responding to security alerts and vulnerabilities, and demonstrate how SBOMs allow consumers to share responsibility and play a more active role in the process.

Table 1: Roles and Responsibilities for Software Vendors and Consumers **without SBOM**

	Vendor	**Consumer**
Box Software	• Monitor components for vulnerabilities • Issue advisories • Author and distribute patches to first-party components • Apply and distribute patches to third-party components • Release patches	• Monitor advisories • Apply, test and rollout patches
SAAS	• Monitor components for	• Monitor advisories

	vulnerabilities • Issue notices • Apply, author and distribute patches • Release patches	

Table 2: Roles and Responsibilities for Software Vendors and Consumers **with SBOM**

	Vendor	**Shared**	**Consumer**
Box Software	• Issue advisories • Author and distribute patches to first-party components	• Monitor components for vulnerabilities • Apply and distribute patches to third-party components	• Monitor advisories • Apply, test and rollout patches
SAAS	• Monitor components for vulnerabilities • Issue notices • Apply, author and distribute patches • Release patches		• Monitor advisories

The production and consumption of accurate SBOMs can allow a consumer to respond faster and more effectively to security events in Box software, because they are already *empowered*, and in some cases, *responsible* for applying these fixes. For SAAS software, the suppliers are already responsible for the entire lifecycle of responding to security issues, so detailed SBOMs are of little use to consumers who don't need to act on the information. Additionally, the feasibility of generating SBOMs for continually changing SAAS software is unclear, since any single SBOM would reflect only one point in time. **It's important to consider the role and responsibility of each party for a particular piece of software in determining how much detail is reasonable and useful in an SBOM.**

Minimum Components for Use Cases

In this table we group the benefits identified by the NTIA's multistakeholder process based on the two primary use cases, then outline the data required to enable these use cases with an SBOM. Some of this data already exists in other formats and databases (ex. NIST's National Vulnerability Database, or detailed FedRAMP compliance reports). In these cases, an SBOM should instead reference these canonical data sources rather than duplicating them.

Usage Type	NTIA-Identified Benefit	Data Required
Vulnerability Lifecycle Management	End-Of-Life	Software Identifiers containing Distributor, Software Package Name and Software Version
	Monitor Components for Vulnerabilities	Software Identifiers that can be joined to a vulnerability source, such as CPES in the National Vulnerability Database
	Identify Potentially Vulnerable Components	Software Identifiers that can be joined to a vulnerability source, such as CPES in the National Vulnerability Database
More Informed Purchasing	Make Code Easier to Review	Software Identifiers containing Distributor, Software Package Name and Software Version
	A More Targeted Security Analysis	Metadata containing functionality and intended use-cases
	Market Signal	Ongoing compliance audits and certification programs, such as FedRAMP or PCI-DSS
	Pre-purchase and Pre-installation planning	Software Identifiers that can be joined to a vulnerability source, such as CPES in the National Vulnerability Database

Guidance is needed from the NTIA on what granularity of data should be presented in an SBOM. More data is not always better. Dependency hierarchy information (transitive vs. direct, etc.) may be useful in some cases, but caution is needed to prevent over-complicating the datasets. Any increase in expressiveness in SBOM formats results in downstream complexity to SBOM tooling for both producers and consumers of the underlying data. **We recommend NTIA work with the SBOM communities to better define requirements on the maximum granularity, depth, and level of detail required to fulfill the intended use-cases.** Several recommendations follow.

Recommendation 1: Simple, Flat SBOMs for Packaged Software

One suggestion is to define simpler, flat versions that are sufficient for these two primary use-cases, with compliant examples from each of the major SBOM formats. For most use-cases, a flat list of dependencies deemed critical enough to monitor, alert, and issue advisories for, along with references to the NVD, can be even more useful than a full graph data

structure. Focusing on fewer, simpler formats will simplify consumption and speed adoption significantly. More complex versions may still make sense for other use-cases in the future.

Recommendation 2: Functional Specs (like FedRAMP) for SaaS

For SAAS software, many of the SBOM use-cases may be better addressed with enhancements to existing Federal requirements, such as FedRAMP. The DevOps Research Agency (DORA)'s 2019 State of DevOps Report found that elite-performing organizations deployed on-demand, multiple times per day. Modern development techniques such as continuous deployment and canary rollouts may mean that there *is no canonical* SBOM for a dynamically changing web service. Rather than including SaaS use-cases in the SBOM requirements, we recommend strengthening the existing programs (such as FedRAMP) through **capturing higher-quality *process* metadata, and including supply-chain hygiene and practices.**

Other Considerations

Data is only useful if it can be trusted. To reach critical mass outlined in the *Amplified "Herd Immunity"* section of the NTIA's paper, SBOMs need to deliver value beyond that already available by static analysis tools that operate on already-built binaries. To deliver this value, the data must be trustworthy, which depends on the method of generation. Guidance is needed from the NTIA on the proper mechanisms to capture trustworthy, verifiable data throughout the supply-chain.

Connect with the larger supply-chain problem. SBOMs will be a valuable resource in helping to identify software contents, but they address only one piece of the open-source supply chain. In the larger picture, build systems and package managers will need to be instrumented to generate "intermediate SBOMs" (intended for *producers* rather than consumers) that attest to the integrity of each step in the supply chain, with tooling to "materialize" these on demand into full SBOMs with the required data for the final consumer. Additionally, there needs to be automated consumption of SBOM data that allows policy checking according to the data they capture. One possibility to address both these considerations is collaboration with SLSA, a security framework that guides software developers on how to securely capture intermediate metadata. In the near future, SLSA plans to add support for automating detailed supply-chain metadata for consumption by policy checkers.

Capture more than the dependencies. Using systems like In-Toto to generate and store cryptographically-verifiable *Attestations* throughout a build process is one technique to preserve verifiable metadata about the entire supply chain. This includes the *how* and the *why* of each step, in addition to the *what* that an SBOM is focused on. Verifiable build can also

address other supply-chain threats, such as post-hoc forensic analysis, or active remediation during an ongoing attack.

Prepare for automation and tooling. Producing and consuming SBOM data with currently available technology requires extensive manual effort; automation will be key to widespread adoption. After this initial phase of recommendations, we recommend focusing on defining more detailed metadata formats that will enable high-quality tooling and automation. Standardized formats that support easy-to-use tooling will be crucial to universal implementation.

Consider the needs of Open-Source Software. OSS is a critical component of our digital infrastructure, so any additional requirements on our supply-chains will inevitably affect the way open source code is produced and consumed. Unfortunately, much of the burden of maintaining our digital infrastructure falls on the backs of unpaid, volunteer contributors. The NTIA should carefully evaluate ways to fund and assist these communities as they work with industry to comply with new regulations.

Conclusion

The mission of SBOMs is to provide valuable supply-chain metadata that enable software producers and consumers to quickly respond to new security threats. In order to meet these goals, the data requirements will need to be carefully considered for the different types and delivery mechanisms of each piece of software. A high signal-to-noise ratio will be critical to adoption, so guidance is needed from the NTIA to ensure that SBOMs contain only the data that is relevant and useful for this intended purpose. This is just the start in securing our national cyber infrastructure. We look forward to continuing to work with the NTIA on leveraging newer technologies that allow for a much richer view into our supply-chains, including data on how software is built and produced.