BLACK DUCK®

GUIDE

WHICH OF CISA'S SIX TYPES OF SBOMS ARE RIGHT FOR YOU?

THERE'S MORE TO SOFTWARE BILLS OF MATERIALS (SBOMS) THAN MEETS THE EYE

While the industry has long agreed on the minimum content of an SBOM, the Cybersecurity Infrastructure and Security Agency (CISA) has classified six specific types of SBOM. For the most part, each SBOM type is created during a particular phase of the software development life cycle (SDLC) and reflects the state of the software as it exists in that moment in time.

At its most basic, an SBOM is a list of ingredients that a piece of software is composed of. One reason that CISA defined these SBOM types is to provide organizations with more information about the state of their software composition at these different moments in time. But this now entails figuring out which SBOM to build, in which situation, and for which purpose.

In this guide, the term "software" should be understood to mean a fully functional application, and the SDLC is used as a rubric for discussing these six CISA-defined SBOMs.

DEFINING THE SIX TYPES OF SBOMS

Here are some quick descriptions of the different types of SBOMs and the benefits and limitations of each type. We're using the SDLC as an organizing principle, but it's important to remember that some SBOM types may be useful across multiple life cycle phases, while others may be appropriate in only one phase. Also, the data presented within any SBOM type may vary, depending on the software's life cycle phase and industry.

Design SBOM

- The Design SBOM covers intended, planned software projects or products with components for new software artifacts—some of which may not yet exist. This information is typically derived from design specifications, RFPs, or initial concepts, and is compiled manually.
- This SBOM type is often missing a number of dependencies that will eventually be included in the final application. But it helps teams plan their course of work by addressing potential issues early.

Source SBOM

- The Source SBOM is created directly from development environment source files and dependencies used to build a product artifact. Typically, it is generated from software composition analysis (SCA) tooling, with manual clarifications.
- This SBOM type is created without visibility into the fully built or running application, and it therefore may lack later life cycle dependencies or even include irrelevant ones.

Build SBOM

- The Build SBOM is generated as part of the build process from data such as source files, dependencies, built components, and ephemeral build process data. The creation of this SBOM type is fully automated as a part of the build phase, following regular configuration actions. This SBOM includes all available elements, including third-party SBOMs, source files, code, and built components, and it integrates intermediate Build and Source SBOMs.
- Because it includes dependencies beyond source code, this SBOM accurately represents the elements of the deployed item. SBOMs created at this stage include other SBOMs, so they may integrate intermediate Build and Source SBOMs for a final release artifact SBOM. Teams may also choose to sign the SBOM during this phase to enable a secure delivery.
- Unfortunately, this SBOM requires substantial configuration effort to integrate with build tools. Some teams may need to adjust their build process to achieve this.

Analyzed SBOM

- The Analyzed SBOM is generated through the analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after the build. Such analysis generally requires a variety of heuristics. In some contexts, it may also be referred to as a "third-party" SBOM because the analysis of the artifacts is performed with third-party tooling.
- This SBOM type requires a binary analysis tool, which can be either automated or manual. Access to source code or build systems is not required. This type of SBOM is created to establish visibility into software created in-house or verify SBOMs provided by vendors or software producers. It can also find dependencies that are hidden to other SBOM generation tools run at different phases. Because Analyzed SBOMs are dependent on heuristics and context, they can be prone to version number inaccuracies or omissions.

Deployed SBOM

- The Deployed SBOM provides an inventory of software that is present on a deployed system. This may be an assembly of other SBOMs that combines analysis of configuration options and an examination of execution behavior in a (potentially simulated) deployment environment.
- This type of SBOM is compiled by manually examining the software installed and present on a system. Because this requires a manual effort, teams will have to take into consideration the information provided by SBOMs and configuration information of artifacts, and then execute the application behavior. This can be done by the software provider in a simulated environment or by the operator in a real or simulated environment.
- Deployed SBOMs can include context of where the software is actually running, but it can be difficult to accurately and completely obtain this information, and many dependencies may reside in inaccessible code.

Runtime SBOM

- The Runtime SBOM is generated by instrumenting the system running the software to capture components present in the system as well as external callouts or dynamically loaded components. In some contexts, this may also be referred to as an "instrumented" or "dynamic" SBOM because it is usually generated by using a dynamic analysis tool, which performs "black box" testing on running applications.
- This SBOM type cuts through the noise and gets to the root of which dependencies should be prioritized for evaluation. Performing this analysis on a running application requires a lot of overhead, and it may take a long time and many test cases to expose all the application's functionality. To make this SBOM reliable and accurate, you must explore every deep, dark corner of an application, which can be difficult without knowledge of the application's architecture.Determining which SBOM type is best for you

Generally speaking, the Build or Analyzed SBOM helps teams hit the intersection of accuracy and efficiency. SBOMs are meant to reveal software components to help identify risk in application dependencies, so accuracy is paramount to both builders and consumers.

Build SBOMs are often the preferred choice for software builders. They allow SBOM generation to be automated with direct integration into the SDLC to build accurate SBOMs throughout the life cycle of every artifact version.

Analyzed SBOMs can also be generated by builders to provide a better, more refined picture of what they're shipping to the consumer. And consumers can generate Analyzed SBOMs on their own to get a reliable picture of application composition, without access to source or build details. This means that builders and consumers can collaborate on the results of an Analyzed SBOM and discuss the deltas if need be.

Build and Analyzed SBOMs also help meet most industry requirements. When combined, these SBOMs include open source dependencies, proprietary code, base images, firmware, operating systems, and any third-party libraries an application may need. Since these SBOM types require tooling and automated generation, you can customize how and when they're generated, as well as specify which fields to include, which formats to generate, and when to generate the SBOM.

In addition, the Build and Analyzed SBOMs enable you to align with the National Telecommunications and Information Administration (NTIA) minimal SBOM requirements, which has become the de facto standard for SBOMs, even outside of the public sector. This means that if you're a software producer, you can satisfy customer requirements. And if you're a software consumer, you can begin to define clear requirements of your vendors, and start establishing control of your own software supply chain.

GETTING STARTED WITH SBOM MANAGEMENT

With this understanding of the types of SBOMs, the question remains: How do you manage all of them?

Essential tooling

Good SCA tooling can help. Organizations have many requirements for SBOM generation and consumption, and determining how to prioritize them can be a challenge. Since SBOMs provide software supply chain visibility to help identify and manage application risk, onboarding an SCA tool can help you establish visibility and map it to that risk.

The most thorough SCA tools can discover dependencies in applications, source code, files, build artifacts, container images, libraries, firmware, and more. Although SCA is mainly used to detect open source dependencies, some tools allow teams to develop and recognize proprietary or commercial dependencies as well. The result of this analysis is a complete SBOM.

SCA tools also provide data sources to enable teams to link dependencies to risk, so they can be assessed for three primary risk considerations.

- Security
  - Does it surpass vulnerability severity thresholds?
  - Does it align with OWASP/SANS compliance?
  - What is the exploitability, fixability, and reachability of vulnerabilities?
- Compliance
  - What are the obligations of each license?
  - Are any licenses in conflict with how the end application will be licensed?
  - Are any licenses on the approved or forbidden list?
- Component health
  - Does the component have active contributors?
  - What is the security reputation of the component?
  - Is this the most recent version of the component?

ESTABLISHING A PROCESS

Many software consumers produce software for their own customers and are obligated to provide an SBOM. While this can be done manually, it's a best practice to incorporate SBOM generation into build systems and use APIs to automate the retrieval of SBOMs that are regenerated with every modification or new build. This produces SBOMs that are machine-readable, which in turn, makes it possible to import the SBOMs back into an SCA tool. This allows you to promptly and continually evaluate dependencies for security, license, and quality risk.

Finally, you should view SBOM creation as a process, not just a document. Any individual SBOM will list application ingredients, but treating SBOM creation as an approach enables dynamic supply chain visibility and upstream risk management.

Treating SBOMs as a process means you should

- Focus on what to include, how often to generate an SBOM, and which technologies to utilize for SBOM administration.
- Learn how to import SBOMs. They can be imported into application security posture management tools, SCA tools, or even a database—anything that enables you to aggregate multiple SBOMs and eventually associate dependencies back to risk for all applications in your portfolio.
- Require actionable SBOM consumption. Many organizations request SBOMs from their software vendors but fail to assess or use them to mitigate risk.
- Assign your own chain of custody for SBOMs. A chain of custody creates a certification mechanism that serves as a verifiable transcript of a product's life cycle and journey.
- Share SBOMs safely with important stakeholders and protect their integrity throughout the chain of custody.
- Enable SBOM search and query. This enables you to comprehend your exposure when the next celebrity vulnerability makes headlines.

How Black Duck can help

When it comes to SBOM generation and management, there is no "one size fits all" approach or solution. Different teams will always have different resources to leverage and different risk appetites, which impacts the type of SBOMs that are required or can be generated. This can understandably lead to confusion and make it hard for teams to know where to start. By focusing on the essential building blocks of the SBOM management approach mentioned in this document, teams can begin moving in the right direction and leverage the momentum to build a complete and tailored strategy.

Black Duck® offers a suite of tools and services that satisfy the essentials of SBOM management, and can give teams the jumpstart they need. Our SCA tools identify application dependencies, generate first-party SBOMs, import third-party SBOMs, surface dependency risk, and guide remediation, all within a single user experience. This way, software builders and consumers alike can establish supply chain visibility and take steps to identify and mitigate risk.

Black Duck recognizes that tooling is only one part of the solution, though. To help our customers transform an SBOM from a document into an efficient process, we leverage our expertise and consulting services to understand a customer's unique situation and help stand up a complete SBOM management strategy.

Learn more

ABOUT BLACK DUCK

Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.