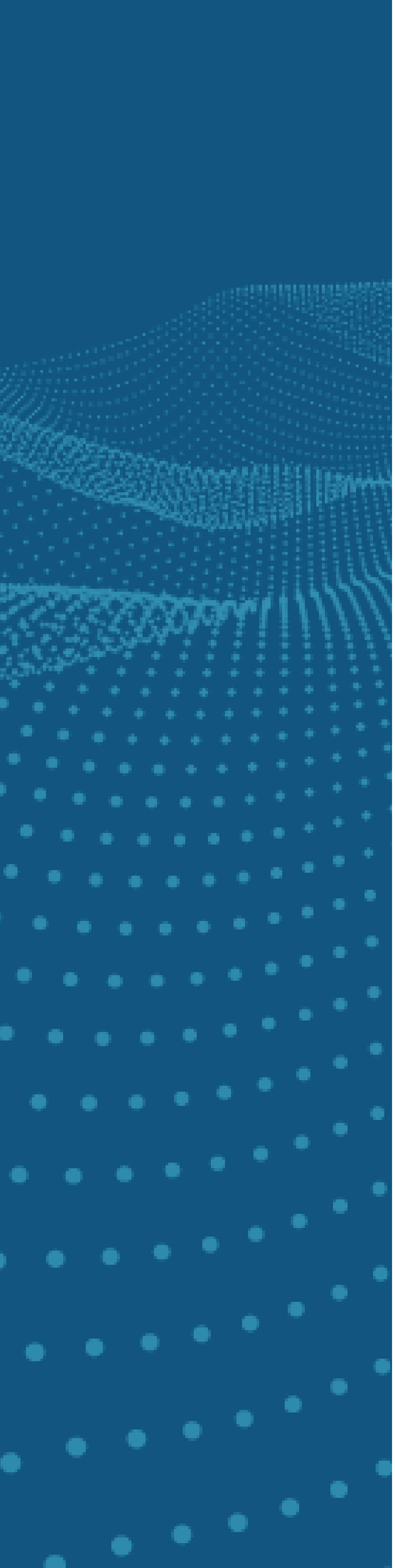

GUIDE

Your Recipe for an Actionable SBOM

Key considerations for building, maintaining, and using a Software Bill of Materials

The list of ingredients in your favorite chocolate chip cookies includes things like flour, sugar, butter, chocolate chips, and so on. But those chocolate chips are made of a subinventory of ingredients—sugar, chocolate, cocoa butter, nonfat milk, and more.

Now imagine if the chocolate chips contained tainted cocoa butter, and a recall was issued for any product containing it. The cookie manufacturer is responsible for those chocolate chips, and it must protect customers from eating bad cookies.

Similarly, software manufacturers need to know the components in their software applications, to protect themselves as well as their end users from components that contain security vulnerabilities or software licensing issues. But without a complete, dynamic view of what's in your applications, neither you, your consumers, nor your vendors can confidently locate or identify a component that might expose you to risk.

What is an SBOM

A Software Bill of Materials (SBOM) is the "list of ingredients" for a software application. It includes all the open source, proprietary, and commercial code, as well as the associated licenses, versions, and patch statuses.

By providing real-time visibility into all components of a software application, an SBOM can help identify risks and threats before bad actors can exploit them. SBOMs can be used by development teams to evaluate the quality, safety, and dependability of their software. Thus, even though SBOMs can't directly prevent attacks, they are cornerstones of successful software supply chain security programs.

But to be effective, an SBOM can't be created once and then forgotten about. Software is constantly changing; new dependencies are added, old ones are updated or removed, and vulnerabilities are discovered and patched. Without continuous updates, an SBOM can quickly become outdated and inaccurate, rendering it useless. Maintaining a comprehensive, up-to-date SBOM helps organizations take proactive measures to mitigate risks and maintain supply chain security and regulatory compliance.

Who needs an SBOM

If you build or consume software, you should care about SBOMs. By generating an SBOM, your development team arms itself and its consumers with information crucial to better understanding the risk associated with a particular application. From there, you're better situated to avoid and react to security breaches and policy violations, or to fix broken, dangerous, or otherwise faulty software. Organizations concerned about software security and quality are including SBOMs as part of their best practices across their software development life cycles (SDLCs).

With nearly 100% of commercial codebases containing open source software, the supply chain is more complicated and obscure, and involves more links and dependencies, than ever before. The only way to mitigate risk is to maintain visibility into all components of your software and address the areas of risk as they are identified.

What happens when risk goes unmanaged

By 2025, Gartner predicts that 45% of organizations worldwide will experience attacks on their software supply chain. Because of the dependencies and connectivity, flaws and vulnerabilities in applications create risk for organizations several degrees away from the initial attack vector. Some prominent attacks that have already occurred include

- **Log4J (2021):** The Apache Log4J critical vulnerability allowed attackers to execute arbitrary code on vulnerable servers, earning a 10/10 on the NVD CVSS severity scale.
- **SolarWinds (2019):** By inserting malicious code into the SolarWinds Orion platform used by 30,000 organizations around the world, hackers gained access to accounts and users without detection. Victims included several Fortune 500 companies and U.S. government departments.
- **Equifax (2017):** Hackers entered Equifax through its customer complaint portal, using a known vulnerability in the Apache Struts framework that had not been patched.
- **Heartbleed (2014):** A bug in the widely used OpenSSL encryption software tricked computers into transmitting the content of a server's RAM, giving hackers access to passwords and personally identifiable information. Victims included Google, Dropbox, Reddit, Facebook, the Canadian Revenue Agency, and many others.

The increase in frequency and severity of cyberattacks on software supply chains has spurred a host of regulations and mandates across industries and governments worldwide. As a result, SBOMs are increasingly mandatory and considered critical to most organizations' regulatory compliance strategy. Regulations and standards requiring or recommending SBOMs include

United States

- Executive Order 14028 directs the National Institute of Standards and Technology (NIST) to develop guidelines for creating and publishing SBOMs and establish criteria for using SBOMs in federal procurement processes.
- The Cybersecurity and Infrastructure Security Agency (CISA) recommends using SBOMs as part of its guidance for secure software development.
- The National Telecommunications and Information Administration (NTIA) defines a set of minimum elements that should be included in an SBOM.

European Union

- The EU Agency for Cybersecurity (ENISA) published the *Guidelines for Securing the Internet of Things*, which recommends providing SBOMs for IoT devices.

United Kingdom

- The U.K. National Cybersecurity Centre (NCSC) recommends that organizations use SBOMs to understand risk and manage vulnerabilities in the software components they use.

Australia

- The Australian Cybersecurity Centre (ACSC) *Information Security Manual: Guidelines for Software Development* recommends using SBOMs to enhance cyber supply chain transparency for consumers.

Canada

- The Canadian Communications Security Establishment (CSE) *Recommendations to Improve the Resilience of Canada's Digital Supply Chain* urges the use of SBOMs to improve transparency and the ability to respond to security attacks.

Clearly, creating and maintaining an SBOM is a critical best practice for building secure and compliant software applications. But how do you get started? And what do you do once you've created it? To help answer these questions, we've compiled several key recommendations.

Recommendation 1

Make SBOM creation a repeatable process

Traditionally, an SBOM is an inventory of the components within a software application. But an SBOM extends beyond a static list; it also encompasses the processes an organization uses to inventory its software.

We encourage organizations to consider SBOMs in the broader sense as a management system. The practices, processes, and activities involved in creating and maintaining an SBOM should be standardized so they are predictable and repeatable. This includes identifying the methods to build SBOM generation into your application development pipelines, deciding when an SBOM should be generated (e.g., every release or commit), and automating SBOM generation by integrating with build tools and repositories.

When considering SBOM solutions, automation is key. NTIA-compliant SBOMs need to be all-encompassing and machine-readable. It's nearly impossible to gather this level of detail manually. Additionally, software builders need a solution that can scale with them—another impossibility with manual processes.

A powerful software composition analysis (SCA) tool can easily generate a complete open source SBOM that includes third-party custom components. Most importantly, SCA tools provide SBOM information on a continuous basis, providing the most complete picture of risks in real time. The right SCA tool can add SBOM generation directly into your SDLC, making the process even easier.

Recommendation 2

Generate SBOMs in standardized formats

Currently, there are three standard SBOM formats: Software Package Data Exchange (SPDX), CycloneDX, and Software Identification (SWID).

- **SPDX** is an ISO standard (ISO/IEC 5962:2021) supported by a rich ecosystem of open source tools and commercial providers.
- **CycloneDX** started in the OWASP community for use in application security and supply chain component analysis.
- **SWID** is supported by NIST and defined by the ISO/IEC 19770-2:2015 standard.

We urge organizations to adopt one of these standards to help their software development teams share metadata about the components in their software packages. Using a common data exchange format also makes it easier to automate the process of generating SBOMs.

Recommendation 3

Become familiar with the different SBOM types

There are a variety of SBOM types. Some types are available and useful across multiple phases of the SDLC, while others are available in only one phase. The data included within an SBOM type may also vary depending on the SDLC phase and industry.

SBOM types include

- Design
- Source
- Build
- Analyzed
- Packaged
- Deployed
- Runtime

We strongly recommend that organizations familiarize themselves with the different types, including their benefits and limitations. Runtime SBOMs, for example, provide visibility into what's in use when systems are running, including dynamically loaded components and external connections. But they may require additional overhead and take more time to create.

We also recommend that organizations use the same type of SBOM consistently. And be cognizant that different types of SBOMS present data differently, so it will not look the same across SBOMs.

Recommendation 4

Plan a trustworthy method of delivery

Organizations need to consider how an SBOM will be delivered once it's produced. Delivering an SBOM in an unsecure or uncontrolled manner could lead to inadvertent or malicious tampering. Planning a secure delivery method helps ensure the security, integrity, compliance, reputation, risk mitigation, and transparency of the software supply chain.

It's also vital that a specific SBOM maps to the correct version of the application. Because an SBOM is associated with a single release of software, each new released version of code requires a new SBOM. Confusion can easily occur in organizations that have dozens of applications with multiple versions. Care must be taken to include component names and version strings among baseline elements of any SBOM.

Recommendation 5

Ensure your SBOM covers your industry's specific requirements

Without a standard definition of what an SBOM should include, organizations previously took it upon themselves to decide how much information the purchaser or end user needed to know about the software components within an application. Similarly, many organizations had to figure out for themselves what to do with their SBOMs once they were generated.

Several regulatory bodies have since stepped in to define the minimum SBOM requirements for certain industries. EO 14028 and NIST have established guidelines for SBOM generation and management for the U.S. government and its vendors, the NTIA has established criteria for healthcare and other industries ("…it is expected that this guidance can be applied to the creation and maintenance of SBOMs in any industry"), and the FDA has established additional criteria for connected medical devices.

The NTIA has defined three minimum elements for SBOMs.

- **Data fields:** These include supplier name, component name, version of the component, other unique identifiers, dependency relationship, author of SBOM data, and timestamp.
- **Automation support:** This is vital for scaling SBOMs across the software ecosystem and organizational boundaries. It requires SBOMs be conveyed in one of the three standard formats (SPDX, CycloneDX, SWID).
- **Practices and processes:** These include frequency, depth, known unknowns, distribution and delivery, access control, and accommodation of mistakes.

The FDA requires additional elements for software in medical devices, citing the heightened need for completeness and accuracy. In addition to the data fields in the NTIA standard, FDA-compliant SBOMs must include support level, support end date, and known security vulnerabilities. These additional elements go beyond open source projects and largely apply to third-party/commercial components within applications.

In the automotive industry, ISO 5230/OpenChain requirements direct SBOMs to include software licensing information.

The key takeaway is that it's not enough to simply point a tool at a codebase and expect the SBOM generated to be sufficient for your needs. You must ensure that your SBOMs are meeting all the specific requirements for the industries in which you do business. Failure to do so could result in penalties, audits, or even suspension of market authorization. Manufacturers risk being excluded from government procurement contracts and partnerships, leading to disruption in supply chains and loss of business opportunities. Noncompliance with SBOM requirements can also tarnish an organization's reputation and erode trust among customers, partners, and regulatory agencies.

Recommendation 6

Include all code in your SBOM

One misconception about SBOMs is that they are only for open source and third-party software components. Going back to the "list of ingredients" analogy, this would be akin to listing only the subinventory of ingredients in chocolate chip cookies. To be comprehensive, an SBOM needs to include ALL components in a software application.

Your SBOM should include

- Proprietary code
- Commercial software development kits
- Private commercial libraries
- Contracted software
- Commercial off-the-shelf code
- Open source software

Less is not better in the case of SBOMs. It's easier to remove elements for certain customers than it is to add elements from different sources. For example, the FDA requires many fields in an SBOM, whereas a specific hospital may require only a subset. We recommend you focus on producing comprehensive, up-to-date SBOMs that you can modify by removing unnecessary elements for special cases.

Recommendation 7

Know what you'll do with your SBOM

Now that you have generated an SBOM, what are you going to do with it? How are you going to map it to areas of risk? How are you going to use it as a tool to control and manage your software supply chain? Answering these questions will ensure that you derive full benefit from your SBOM to reduce supply chain risks, increase customer confidence, and support regulatory compliance.

An SBOM can help identify risks and threats before bad actors can exploit them. But to be effective, it can't be created once and then forgotten about. Software is constantly changing; new dependencies are added, old ones are updated or removed, and vulnerabilities are discovered and patched. Without continuous updates, an SBOM can quickly become outdated and inaccurate, rendering it useless. By following these seven recommendations, your organization can generate, manage, and use SBOMs to build secure and compliant software applications.

Learn more about generating, managing, and using SBOMs