Roles and Benefits for SBOM Across the Supply Chain

Use Cases Working Group

This document was drafted by an open working group convened by the National Telecommunications and Information Administration in a multistakeholder process, including the following individuals and organizations: $Org1; $Name2, $Org2; $Name3, Org3... Others participated, but do not wish to be named. Input into this document included numerous interviews, creation of use cases, and input from the Healthcare PoC..

Introduction

Software is everywhere. Like steel and concrete, software increasingly plays a foundational role in modern, connected society and like those other building materials, how and with what ingredients the building materials are created often matters. Software permeates banking, healthcare, utilities, emergency services, national defense, government systems, and the like. Software includes operating systems, firmware, and embedded systems within our gadgets, devices, IoT, and other machines. And just like these physical goods, software too has a supply chain that may need to be understood and managed by an organization dependent on that software.

Most software includes other software. Software changes and evolves over time due to optimization, new features, security fixes, and so forth. As a result, software producers throughout the supply chain have to continually evaluate how changes might impact their software. This includes changes to 3rd-party components used to compose software. How can organizations make confident, informed decisions? How can they manage the complexity of their software supply chain in a sustainable manner? In a complex supply chain, roles can blur. For simplicity, we will initially describe the software supply chain from three perspectives:

- I *produce* software - the person/organization that creates a software component or software for use by others [write/create/assemble/package]
- I *choose* software - the person/organization that decides the software/products/suppliers for use [purchase/acquire/source/select/approve]
- I *operate* software - the person/organization that operates the software component [uses/monitor/maintain/defend/respond]

This document describes the benefits of having a Software Bill of Materials (SBoM) from all three perspectives, summarized in the following table:

	Perspective on Software		
Benefit	Produce	Choose	Operate
Cost	Less unplanned, unscheduled work	More accurate total cost of ownership	More efficient administration
Security Risk	Avoid elective vulns	Easier Due Diligence	Faster Identification/ Resolution Am I affected? Where am I affected?
License Risk	Avoid elective license risk	Easier Due Diligence	More efficient, accurate response to complaints
Compliance Risk	Easier Risk Evaluation, Catch issues earlier in lifecycle	More accurate Due Diligence, catch issues earlier in lifecycle	Streamlined process
High Assurance (See Appendix II)	Make assertions about artifacts, sources, and processes used.	Making informed, attack-resistant choices about components.	Validate claims under changing and adversarial conditions.

The Software Supply Chain

Even before software was widespread, organizations thought about multi-stage production processes through the lens of the *supply chain*: each stage of production takes inputs from a previous stage and adds their own skills and contributions to produce outputs that a subsequent stage can use. At one end are the most basic components, such as raw materials, and at the other end are the final users or consumers of the product. Each link in well-functioning supply chain provides an opportunity to ask crucial questions, such as “does this input meet my quality standards?” and “am I using the correct input, or should I use something else?”

It’s useful to think of modern-day software development as a supply chain:

- Software developers write code that fulfills a need, then make it available freely or commercially.
- Other developers with similar needs find that code and include it in their own software.
- At some point, a product manufacturer assembles software components into a product.
- End users acquire and operate the finished product.

The supply chain is a simple model of how products are made, but it doesn’t answer every possible question. What happens when something goes wrong with a link in the chain?

In the world of physical goods, “upstream” parts might be recalled or upgraded, and the relationships in the supply chain might be strong enough to make sure that any necessary changes are made “downstream.”

In the world of software, the links between dependencies are weaker. Sometimes, there is no direct commercial relationship between the links. And unlike most physical parts, software components change constantly. Every participant in the software supply chain is continually weighing choices and grappling with changes introduced by the release of new software versions, security vulnerabilities, etc. Because of the complex web of dependencies in software supply chains, any change can have far-reaching effects.

About this document: Goals and Methodology

This document was drafted by the Use Cases Working Group as part of NTIA’s multistakeholder process on software component transparency.[1] The group was initially driven by several observations. First, participants noted that SBoMs (and things like them) were already in use today, and thus the existing use cases should be documented and captured. Second, participants embraced the idea that this was a supply chain problem, and any transparency solution should address the broader software supply chain, rather than any single subset or sector. Third, stakeholders argued that since the benefits of transparency accrue through different mechanisms across the supply chain, they should be carefully mapped out. A key aspect of this work was to understand how the efforts of the broader NTIA initiative could impact the software ecosystem and increase awareness and adoption for the growing community of those interested in SBoM practices.

[1] Link to other SBoM docs

The group set out with two goals: to capture the SBoM use cases today to find out what works and what needs to be improved, and to understand how existing software practices could be improved by wider adoption of SBoMs. Participants wanted to avoid further reinventing the wheel at each organization facing their own software supply chain challenges. As noted in Appendix 1, this work is not happening in a vacuum.

Much of this work was inspired by existing industrial and supply chain work.[2] We have adapted that framing to better match the structure of the software industry today and make the lessons learned more accessible. The group met with weekly calls to extract knowledge and debate these use cases. The expertise of working group members was supplemented by a series of expert interviews, each lasting at least an hour, meant to focus on how different actors in the software supply chain understood their risks, costs, and the potential value of transparency. The group decided to focus on the points of view of three perspectives (as discussed above) and capture the current or potential benefits that greater software transparency can provide.

This document is supplemented by two appendices. Appendix I contains references to other initiatives that explicitly or implicitly acknowledge the value and importance of transparency in the supply chain. Appendix II contains a discussion of the value from assurance attributes around SBoMs that may be necessary in certain applications. These higher assurance points will be addressed in the future work of the NTIA process.

Perspective: Produce Software

Code reuse is an integral part of modern software. In addition to writing their own code, producers of software integrate third party components into their software. Organizations that make software continually weigh whether to build components from scratch or import components from elsewhere. To keep projects moving at high velocity, this decision making is often diffused among teams and individual developers. Yet consideration of components or ingredients in any product is a key piece to producing a high-quality product.

For many organizations, review and vetting of open-source software they choose to include in their products and services began because of legal concerns about including software made available under restrictive open-source licenses that put limitations on distribution. However, this quickly evolved to include a review of security concerns as well. A bill of materials is integral to understanding the supply chain of any product, and in the software space, it is hard to understand anything about the supply chain risk without visibility into the underlying components.

A Software Bill of Materials (SBoM) can help a software supplier produce their software in the following ways:

[2] See, e.g. Deming's (1986) seminal works on supply chain quality

An SBoM can **reduce unplanned, unscheduled work** by offering better visibility into the broader code base. This can allow a more organized prioritization, reducing costs in time and money. For example, when a new vulnerability emerges, an SBoM allows an organization to move finding the fault upstream in the process, reducing complexity in remediation.

An organization that tracks components can more easily **reduce code bloat.** For example, an organization might use multiple versions of the same component, even loading them into the same executable. SBoMs make it easier for an organization to standardize on a common code base, and reduce complexity by supporting the establishment of a smaller set of moving parts.

More broadly, **making it easier for developers to "zoom out" to understand dependencies within broader, complex projects** can facilitate more responsibility and better quality management. A more systematic approach to code reuse can allow greater confidence in the overall process and therefore product. An organization can better track needed experience/expertise for particular teams or products, make sure that potential future maintenance is supportable, and avoid surprises when downstream customers evaluate software.

It enables an organization to **know and comply with the license obligations** of the components used. A system that makes licensing policies easy to use can help automate license compliance.

An SBoM-equipped producer can more easily **monitor components for vulnerabilities** so the team can more proactively evaluate and remediate risks. When a new security risk is discovered by security researchers, identifying whether or not a particular product is potentially vulnerable can be a drawn out process. An easily accessible list of components can make this process much more efficient. Better awareness of components can also shorten the window to reassure customers, improving customer trust.

Sometimes, software components reach their **end-of-life (EOL)** and are no longer supported by that upstream supplier, or the supplier disappears entirely. A responsible producer should actively monitor for this, and plan for contingencies before they arrive. An SBoM enables an organization to be proactive with their supply chain to identify and implement alternate solutions.

Tracking components and sub-components can **make code easier to review** and understand for developers, simplifying builds and reducing obstacles to **getting code into production**. Much of the initial security testing for avoidable harm can happen in-context, as the code is written/assembled, weeks or months earlier than alternative. Furthermore, this tracking enables more situational awareness when an underlying dependency changes. It can also provide a better understanding of the work and time needed to make a change to the codebase.

Tracking component usage can support strategies like **a blacklist of banned components or a whitelist of preferred components -** or both. Blacklists allow companies to avoid components with known issues, orphaned projects, or that have had a history of having many security issues. Whitelists, while not as common, allow companies to use trusted third party components and invest in their use, or have a list of preferred providers.

An organization can **provide an SBoM to its customers** or downstream partners to help assure them that the company is providing a high-quality product that meets customers' legal and security needs (see below for how an SBoM can help those who choose and maintain software). Being proactive can offer a competitive advantage while SBoMs are being adopted more, and may ultimately become a common market expectation or requirement. An up-to-date SBoM can also reassure downstream consumers about the current security status of a product in their possession.

Perspective: Choose Software

Choosing software includes the selection and acquisition of software products. These processes differ from one organization to another, but while there are many ways of choosing software, there are a few well-known steps common to all of them.

Almost any choice of software is a long-lasting commitment, and it is therefore very important that the decision to acquire one software product or component vs. another is made with forethought and planning. This acquisition process can be formal or informal, and may include a series of steps, such as reviewing requirements, reviewing the market, evaluating suppliers and software products, and the steps needed to actually acquire the software. A number of different roles could be involved, including users (hopefully), finance and legal, and the technical support team.

Consumers of software without an SBom know there is probably open source components inside the code, but the software supplier rarely exposed them. Those ingredients could include unsupported 'orphan' libraries or components with known vulnerabilities. Malicious suppliers could hide malware inside components that performed useful functions.

A Software Bill of Materials (SBoM) can help an organization choose their software and supplying organization in the following ways:

Visibility into underlying components can help **identify potentially vulnerable components**. Prior to purchase, an organization can conduct the basic risk analysis to understand what they are about to put on their systems.

Organizations with a more mature risk process can engage in a **more targeted security analysis** process by deciding what code components might raise red flags (such as a cryptographic library that offers substandard protection) and which components might have already been vetted by a trusted source.

If the SBoM includes hashes of the components, an organization can **verify sourcing** of third party components to limit the risk that counterfeit or backdoored components slipped into the supply chain of the supplier. (While an SBoM may not directly prevent a determined adversary from interfering with the legitimate source, it can greatly simplify remediation once the attack is detected--see above.)

An organizations may face regulations or other rules around supply chain sources, and an SBoM can enable **compliance with policies** around what must or must not be used in their organization.

An organization can be made **aware of end-of-life components** for which future support will not be available. While these components may not be a risk today, the uncertainty about future maintenance can empower better planning for the future.

The acquirer will be better able to **verify some claims** by the supplier about the code base and its quality. While knowing which components and versions will not answer every question about the quality and security of software, it can offer some insight into the supplier's vigilance.

An organization can better **understand the software's integration** into existing asset and vulnerability management systems, lowering the overall cost of ownership and minimizing the likelihood of risks emerging from poor integration.

Finally, a verified SBoM provides a **market signa**l that a supplier is thinking about possible risks from its included components, indicating the supplier is practicing good software development hygiene and observes at least some best practices. This can have the most marked impact on the suppliers by rewarding those who invest in security and quality processes.

Choosing software means choosing a supplier, and any choice of supplier also means inheriting all of its suppliers as a consequence. Transparency ensures this is a more informed choice.

Perspective: Operate Software

Once any software package or component is selected and acquired, it must be installed, configured, maintained, and administered. We group these responsibilities under the category of "operation." They vary for different organizations, and could cover a range of potential roles, ranging from the administrator to the NOC or SOC to an executive in charge of risk or compliance. These roles may also apply for non-IT packages such as embedded software in an industrial or OT setting. we have identified two distinct groups of personnel who will be playing these roles with some examples.

An SBoM can help an organization configure, maintain, and administer its software in the following ways:

A list of components allows for the **easy identification of new vulnerabilities** as they might be discovered over the lifetime of a piece of software. When new potential flaws are identified by 3rd parties, an understanding of what is running on one's network allows an organization to quickly assess whether they are potentially at risk without having to wait for the supplier to communicate. Data on new vulnerabilities can come from any source.

Awareness of underlying potentially risky components can **drive independent mitigations** while an organization waits for their supplier to assess the actual risk and offer software updates as needed. A risk averse organization may choose to take other steps in the interim, such as network segmentation, or tuning IDS/IPS or other network defenses to detect or preclude any potential exploitation.

More broadly, an SBoM allows an operator to **make better risk-based decisions** about what is on their network, and how to prioritize a response, driven by their own approach to security and risk. As several participants have put it, an SBoM offers a **"roadmap for the defender,"** particularly when it comes to vulnerabilities that might be linked to widespread exploitation and automated tools such as metasploit.

Careful understanding of third party components can enable **alerts about potential end-of-life (EOL)** situations. By combining data from SBoMs with other data sources, an organization can understand when a component may no longer be supported by its supplier. This will allow that organization to understand the potential ripple effects for the software using those components, and make proactive decisions to work with their supplier or seek alternatives.

More information about components can **better support compliance and reporting requirements**. In addition to being a more detailed asset inventory, SBoMs support a post-market surveillance requirement or a requirement to follow up on security alerts to demonstrate vigilance.

Documented software components can **reduce costs through more streamlined and efficient administration**. Those responsible can quickly identify points of concern, and would not have to spend time trying to contact suppliers when they can determine via the SBoM that the software does not contain a deficient component.

Ecosystem Perspectives and the Public Value of SBoM

While this document has focused on specific team and organization benefits, there other considerations across the entire software ecosystem. This document will further discuss the public good promises and issues of SBoM in subsequent drafts. For example, SBoMs can support better ecosystem-wide vulnerability disclosure, and greater attention to the risks of widespread use of unsupported software.

Appendix I Related efforts that explicitly or implicitly highlight the value of SBoM

The work of the NTIA initiative does not occur in a vacuum. There are a number of key works across the ecosystem that have advanced or highlighted the importance of an SBoM at various points in the supply chain. The following is a list of related projects that are included to emphasize the growing support and importance of SBoM. It is not an exhaustive list and the projects below are not endorsed by this group.

BSA Framework for Secure Software.
This industry-drafted framework offers guidance on secure development of software, security capabiites of software, and a secure lifecycle, citing standards and other authoritative guidance. It makes repeated references to the importance of tracking third party code, including advising "To the maximum feasible through the use of manual and automated technologies, subcomponents integrated in third party components are documented, and their lineage and dependencies traced."

Building Security in Maturity Model
BSIMM (Building Security in Maturity Model) is a large group of software developers in academia, government, and industry that benchmark best software development practices. They create practices that organizations can benchmark themselves against and assess where they are relative to their peers in their industry or overall. They are up to version 9 and contain several SBoM related requirements:

SR2.4 "Identify open source"
SR3.1 "Control open source risk"
CMVM2.3 "Develop an operations inventory of applications"
SFD3.2 "Require use of approved security features and frameworks"
SE3.6 "Enhance application inventory with operations bill of materials"

CISQ Trustworthy System Manifesto
The Council on IT Software Quality (CISQ) has published the "CISQ Trustworthy System Manifesto" on holding senior executives accountable for cybersecurity. Section III is "Traceable Properties of System Components" which has requirement #2 "Evidence of provenance and trustworthiness should be carried forward with components and shared across the supply chain" which contains in the description:

> "When developers incorporate open source components, external APIs, or microservices, they should document their source and related data for inclusion in a System Bill of Materials (SBOM)."

FDA Premarket Guidance

The US Food and Drug Administration (FDA) has published draft Pre-Market Guidance for medical device manufacturers seeking FDA certification. This guidance maintains that: "The device design should provide a CBOM in a machine readable, electronic format to be consumed automatically" where Cybersecurity Bill of Materials (CBOM) is defined as " a list that includes but is not limited to commercial, open source, and off-the-shelf software and hardware components that are or could become susceptible to vulnerabilities."

FS-ISAC Third Party Governance
The Financial Section Information Sharing and Analysis Center (FS-ISAC) published "Appropriate Software Security Control Types for Third Party Service and Product Providers". It includes Control Type 3B, "A Bill of Materials (BOM) for Commercial Software to Identify Open Source Libraries Used".

ISO

ISO/IEC 27002:2005 and 27002:2013 have relevant sections that highlight the importance of best practices to ensure adherence to information security control objectives for an organization. The perspectives identified in this document can be aligned this international standard to show its relevance to a broad number of industries that use/produce information systems.

Make Software	ISO/IEC 27002:2005, 9.2.6, "Secure Disposal or Re-use of Equipment"; Section 10.4, "Protection against Malicious and Mobile Code"; Section 15.1.2. "Intellectual Property Rights(IPR)." ISO/IEC 27002:2013, Section 14: System acquisition, development and maintenance - 14.2 Security in development and support processes
Choose Software	ISO/IEC 27002:2013, Section 14: System acquisition, development and maintenance 15: Supplier relationships - 15.1 Information security in supplier relationships
Operate Software	ISO/IEC 27002:2005, Section 5.1.1, "Inventory of Assets." and Section 10.4, "Protection against Malicious and Mobile Code"; Section 15.1.2. "Intellectual Property Rights(IPR)."

	Section 9.2.7, “Removal of Property”; Section 9.2.6, “Secure Disposal or Re-use of Equipment”; Section 10.7.2, “Disposal of Media.” Section 13.2, “Management of Information Security Incidents and Improvements”; Section 10.10.2, “Monitoring System Use”; Section 15.2.2, “Technical Compliance Checking”;

Linux Foundation OpenChain
The Linux Foundation OpenChain project is on the use of open source and contains: “A process exists for creating and managing a FOSS component bill of materials which includes each component (and its Identified Licenses) from which the Supplied Software is comprised”

Manufacturers Disclosure Statement for Medical Device Security
The Manufacturer Disclosure Statement for Medical Device Security (MDS^2) was originally developed by the Healthcare Information and Management Systems Society (HIMSS) and the American College of Clinical Engineering (ACCE), and then standardized through a joint effort between HIMSS and the National Electrical Manufacturers Association (NEMA). The MDS^2 form provides medical device manufacturers with a means for disclosing to healthcare providers the security related features of the medical devices they manufacture.

MITRE Deliver Uncompromised
MITRE, in its report on the national security supply chain, “Deliver Uncompromised” recommends SBOM’s as part of supply chain integrity. It notes, “If done properly, an SBOM can estimate the overall risk of the ensemble of software elements based on the risk of the individual elements.”

NIST’s Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)
A 2019 draft white paper recommends core set of high level secure software development practices. Among these, it recommends that organizations seeking to protect their software “Create and maintain a software bill of materials (SBOM) for each piece of software stored in the repository.”

OWASP Component Analysis Project
This industry expert group’s guidance on component analysis recommends “Contractually require SBOMs from vendors and embed their acquisition in the procurement process” and an list of best practices using the SBoM to improve security

SAFECode Managing Security Risks Inherent in the Use of Third party Components
Industry group SAFECode drafted a white paper capturing the collective knowledge on the benefits and challenges of managing third-party code risk in product development.

Software Heritage
Software Heritage is a non profit initiative building a universal archive of software source code, as a common infrastructure catering to a variety of use cases from industry to science and culture.
One of the use cases specifically listed on their mission statement is source code tracking for industry: "*Because industry cannot afford to lose track of any part of its source code, we track software origin, history, and evolution. Software Heritage will provide* ***unique software identifiers, intrinsically bound to software components****, ensuring persistent traceability across future development and organizational changes.*"

Appendix II - Assurance and Confidence Use Cases and Elements of an SBoM

The basic SBoM described in this document and related efforts can offer many benefits, as discussed above. However, some use cases may require more information about the software or the SBoM itself. Beyond the components, the producer, selector, or operator may want to know more about the components and their creators, how the components were assembled, or how the SBoM itself was compiled. Organizations that would suffer dire consequences from allowing maliciously altered or contaminated software may want these further elements.

Provenance of an SBoM is the term of art for having information about the chain of custody of the software and all of the constituent components that comprise that software, capturing information about the authors and locations from where the components were obtained. Whether a component comes directly from the supplier's distribution site or some other location can be a concern for some organizations. Similarly, understanding the exact identity of the supplier can help an organization establish where to go for updates or to communicate about bugs or enhancements. Finally, access to authorship allows organizations to correlate their experience with components to the creators and rank their internal preferences through reputation-like scoring of providers of software.

Pedigree of an SBoM is the term of art for having information on all of the components that have come together to make a piece of software and process under which they came together. This can include details beyond components, such as compiler options. For example, understanding whether compilation options invoking ASLR were used or not used indicates that the resultant piece of deployable code is hardened against certain types of attacks. Understanding of the process used in taking the source code and incorporated components and libraries to formulate the resultant executable is an important source of insight for those who need to know what selection of options were used in creating the executable software.

Integrity of the SBoM refers to the use of cryptographic techniques to indicate that the SBoM hasn't been altered since written by its author or if there was a modification it indicates that alteration by some subsequent SBoM author. Being able to determine the SBoM's integrity can help, for example, in situations where there is concern about whether an adversary may be purposefully trying alter the SBoM to mislead those using them for analysis of vulnerabilities. If someone edits the SBoM to indicate it has a later, non-vulnerable version of a component, the organization will be left susceptible to attacks against that vulnerability even though the altered SBoM indicates they are using a non-vulnerable version. Similarly, alteration of the authorship or source information would undermine the Provenance of the SBoM or alteration of the details of the formulation choices would undermine the Pedigree of the SBoM.

These three SBoM features can supplement the benefits above to provide concrete security benefits, particularly for organizations that face active threats to their supply chain from determined adversaries. Future work will explore these potential SBoM use cases, and map them to specific elements.