NOTE: Comments from the Cybersecurity and Infrastructure Security Agency are highlighted in the margins and in-line.

DEPARTMENT OF COMMERCE

National Telecommunications and Information Administration

[Docket No. 210527–0117] RIN 0660–XC051

Software Bill of Materials Elements and Considerations

AGENCY: National Telecommunications and Information Administration, U.S. Department of Commerce.
ACTION: Notice, request for public comment.

Background

On May 12, 2021, the President issued Executive Order 14028, ‘‘Improving the Nation’s Cybersecurity.’’ [1] An initial step towards the Executive Order’s goal of ‘‘enhancing software supply chain security’’ is transparency. As the Order itself notes, ‘‘the trust we place in our digital infrastructure should be proportional to how trustworthy and transparent that infrastructure is, and to the consequences we will incur if that trust is misplaced.’’ An SBOM advances transparency in the software supply chain, similar to a ‘‘list of ingredients.’’ NTIA is directed to publish a list of ‘‘minimum elements for an SBOM.’’ NTIA has played a leadership role in advocating for SBOM, convening experts from across the software world and leading discussions around the ideas of software supply chain transparency.[2] The goal of this Request for Comments is to seek input and feedback on NTIA’s approach to developing and publishing the minimum elements of an SBOM. NTIA is committed to being open to further additions, corrections, deletions, or other changes, particularly when suggestions are well supported with documents, operational evidence, and support from broad-based constituencies in the software ecosystem.

Since 2018, NTIA has coordinated an open and transparent multi-stakeholder process on software component transparency, providing a forum in which a diverse and evolving set of experts and interested parties have been able to weigh in, share their leadership and respective visions, unpack the complex challenges of software supply chain, and propose various solutions.[3] The idea of an SBOM is not new. Its roots lie in the concepts developed by noted American engineer and management consultant W. Edward Deming to build post-war industrial supply chain leadership, and over the last decade an SBOM has come to be considered vital to security by notable security experts.[4] By providing a forum for SBOM discussions, NTIA has helped the community identify common themes, coalesce around standards, and emphasize interoperability. These discussions have led to the documentation of existing tools, products, and projects, and have helped drive further experimentation and implementation. With an emphasis on the practice of SBOM generation and use, NTIA has sought to facilitate ‘‘proof-of-concept’’ exercises in specific communities and sectors.[5] NTIA has also worked across the federal government to share ideas about SBOM, seek feedback and engagement from experts in the civilian and national security community, and expand general awareness of SBOM.

What is an SBOM?

The Executive Order defines an SBOM as ‘‘a formal record containing the details and supply chain relationships of various components used in building software.’’ It refers to what the software assurance organization SAFECode calls ‘‘third party components.’’ Software is made and used by a wide range of organizations, but this diversity makes a single model for SBOM difficult. There is no one-size-fits-all approach to providing transparency for software assurance. The Executive Order also defines SBOM in functional terms, framing its value in terms of use cases. It notes distinct but overlapping benefits that accrue to the organization that makes the software (‘‘developers’’), the organization that chooses or buys software, and those that operate software. Many of these use case benefits center around tracking known or newly identified vulnerabilities, but SBOM can also support use cases around license management and software quality/efficiency, and can lay the foundation to detect software supply chain attacks. These benefits should serve as a lodestar for designing and publishing the minimum elements of an SBOM that can be applied across the diverse software ecosystem.

Potential Elements for an SBOM

NTIA proposes a definition of the ‘‘minimum elements’’ of an SBOM that builds on three broad, inter-related areas: Data fields, operational considerations, and support for automation. Focusing on these three elements will enable an evolving approach to software transparency, and serve to ensure that subsequent efforts will incorporate more detail or technical advances. The information below is preliminary, and the ultimate list published by NTIA will be revised based on public input.

Data fields. To understand the third-party components that make up software, certain data about each of those components should be tracked. This ‘‘baseline component information’’ includes: [6]

- Supplier name
- Component name
- Version of the component
- Cryptograph hash of the component
- Any other unique identifier
- Dependency relationship
- Author of the SBOM data

Some of these data fields could be expanded. For example, the ‘‘dependency relationship’’ generally refers to the idea that one component is included in another component, but could be expanded to also include referencing standards, which tools were used, or how software was compiled or built. Other data fields may need more clarity, including data fields for component and supplier name. As one SBOM document notes, ‘‘[c]omponent identification is fundamental to SBOM and needs to scale globally across diverse software ecosystems, sectors, and markets.'' [7] The challenge is that different technical communities and organizations have different approaches to determining software identity.

Operational considerations. SBOM is more than a set of data fields. Elements of SBOM include a set of operational and business decisions and actions that establish the practice of requesting, generating, sharing, and consuming SBOMs. This includes:

- **Frequency.** Operational considerations touch on when and where the SBOM data is generated and tracked. SBOM data could be created and stored in the repository of the source. For built software, it can be tracked and assembled at the time of build. A new build or an update to the underlying source should, in turn, create a new SBOM.

- **Depth**. The ideal SBOM should track dependencies, dependencies of those dependencies, and so on down to the complete graph of the assembled software. Complete depth may not always be feasible, especially as SBOM practices are still novel in some communities. When an SBOM cannot convey the full set of transitive dependencies, it should explicitly acknowledge the ''known unknowns,'' so that the SBOM consumer can easily determine the difference between a component with no further dependencies and a component with unknown or partial dependencies.

- **Delivery**. SBOMs should be available in a timely fashion to those who need them and have proper access permissions and roles in place. Sharing SBOM data down the supply chain can be thought of as comprising two parts: How the existence and availability of the SBOM is made known (advertisement or discovery) and how the SBOM is retrieved by or transmitted to those who have the appropriate permissions (access).[8] Similar to other areas of software assurance, there will not be a one-size-fits-all approach.

  Anyone offering SBOMs must have some mechanism to deliver them, but this can ride on existing mechanisms. SBOM delivery can reflect the nature of the software as well: Executables that live on endpoints can store the SBOM data on disk with the compiled code, whereas embedded systems or online services can have pointers to SBOM data stored online.

- **Automation support**. A key element for SBOM to scale across the software ecosystem, particularly across organizational boundaries, is support for automation, including automatic generation and machine-readability. As the Executive Order notes, SBOMs should be machine-readable and should allow ''for greater benefits through automation and tool integration.'' Manual entry or distribution with spreadsheets does not scale, especially across organizations. The SBOM community has identified three existing data standards (formats) that can convey the data fields and be used to support the operations described above: SPDX,[9] CycloneDX,[10] and SWID tags.[11] Experts in these formats have mapped between them to create interoperability for the baseline described above. Because these formats already are subject to public input and translation tools exist, they serve as logical starting points for sharing basic data.[12]

In addition to the three SBOM formats, the need for automation defines how some of the fields might be implemented better. For instance, machine-scale detection of vulnerabilities requires mapping component identity fields to existing vulnerability databases.

Request for Comment

The discussion above lays out the collected data points and experience from experts and practitioners in SBOM, including existing practices and novel proof-of-concept work. To inform, validate, and update NTIA's understanding of SBOM, NTIA seeks comment on the following questions:

1. Are the elements described above, including data fields, operational considerations, and support for automation, sufficient? What other elements should be considered and why?

Restricted delivery of SBOM material seems like it will present a real challenge for automation. This is especially true when you try to relay information from upstream – i.e., How does supplier A that includes a component from supplier B know what to include about that component in their SBOM? Perhaps there should be an element in the SBOM that explains how the information in the SBOM can be communicated.

Another operational issue is the longevity of the need for SBOM information and whether its availability aligns with that need. ICS devices often need to remain in operation for 10-15 years or more. The likelihood that a web-based delivery system for the SBOM information will remain available for that long seems unlikely. In these cases, it's probably best to save local copies of the SBOM and ingest them into some kind of management system with backups.

1. Are there additional use cases that can further inform the elements of SBOM?

None that we are aware of.

SBOM creation and use touches on a number of related areas in IT management, cybersecurity, and public policy. We seek comment on how these issues described below should be considered in defining SBOM elements today and in the future.

A. **Software Identity:** There is no single namespace to easily identify and name every software component. The challenge is not the lack of standards, but multiple standards and practices in different communities.

This is one of the biggest challenges and extends to names of suppliers as well. A centralized authority seems unrealistic given the breadth of the software industry. Perhaps a federated system could be developed, with SBOMs that identify which naming regime they fall under. Perhaps we could leverage the DNS system for supplier naming although I'm sure there will be challenges.

Another consideration is that versions do not always follow standards. This can make matching published vulnerabilities to specific versions of software difficult. For example, many of the CVEs use version ranges to indicate which versions are vulnerable. If versioning schemes do not adhere to sequential numbering, then it can be difficult to know which versions are affected.

In a similar vein to name conflicts, suppliers will edit open-source libraries and package them alongside their "original" code. Any addition in the referenced library will change the hash.

B. **Software-as-a-Service and online services:** While current, cloud-based software has the advantage of more modern tool chains, the use cases for SBOM may be different for software that is not running on customer premises or maintained by the customer.

In these cases, it is more important to understand the interfaces and the data that passes through the SaaS offering. This may be a potential extension to SBOMs.

C. **Legacy and binary-only software:** Older software often has greater risks, especially if it is not maintained. In some cases, the source may not even be obtainable, with only the object code available for SBOM generation.

Archaeological SBOM generation tools can help, but they are only as good as the information in the reference database. If the binary software is obscure enough then we may not be able to provide much additional information beyond the file hash and supplier and name (if those are even known)

Integrity and authenticity: An SBOM consumer may be concerned about verifying the source of the SBOM data and confirming that it was not tampered with. Some existing measures for integrity and authenticity of both software and metadata can be leveraged.

This is an important consideration, especially when working to detect supply chain attacks. If the software build system is also responsible for generating the SBOM then what is to stop an attacker from modifying the SBOM to remain consistent with the trojanized software?

D. **Threat model:** While many anticipated use cases may rely on the SBOM as an authoritative reference when evaluating external information (such as vulnerability reports), other use cases may rely on the SBOM as a foundation in detecting more sophisticated supply chain attacks.

These attacks could include compromising the integrity of not only the systems used to build the software component, but also the systems used to create the

SBOM or even the SBOM itself. How can SBOM position itself to support the detection of internal compromise? How can these more advanced data collection and management efforts best be integrated into the basic SBOM structure? What further costs and complexities would this impose?

One option would be a trusted third party that is responsible for generating the SBOM given the source and/or binary from the supplier. This would make it more difficult for a supply chain attack to affect the SBOM. However, a centralized trusted authority would also be an ideal target.

E. **High assurance use cases:** Some SBOM use cases require additional data about aspects of the software development and build environment, including those aspects that are enumerated in Executive Order 14028.[13]

How can SBOM data be integrated with this additional data in a modular fashion?

Relationship information such as "Compiled with" or "Built with" can be included and is already mentioned above as a potential for expansion of the proposed data fields.

F. **Delivery:** As noted above, multiple mechanisms exist to aid in SBOM discovery, as well as to enable access to SBOMs. Further mechanisms and standards may be needed, yet too many options may impose higher costs on either SBOM producers or consumers.

G. **Depth:** As noted above, while ideal SBOMs have the complete graph of the assembled software, not every software producer will be able or ready to share the entire graph.

There should be some way for SBOM producers to indicate what depth their SBOM covers. In the case of the Longclaw tool (a software assurance tool developed through a CISA partnership with a National Laboratory to conduct binary and code analysis on software and firmware inputs), we deliver a "flat" SBOM given than we are not easily able to determine subcomponent dependencies since all statically linked code looks the same. We are working on improvements based on known origins of functions and dependency graphs so we may be able to eventually offer more depth in our SBOMs

H. **Vulnerabilities:** Many of the use cases around SBOMs focus on known vulnerabilities. Some build on this by including vulnerability data in the SBOM itself. Others note that the existence and status of vulnerabilities can change over time, and there is no general guarantee or signal about whether the SBOM data is up-to-date relative to all relevant and applicable vulnerability data sources.

Vulnerability information is best mapped outside of the SBOM, but should absolutely leverage information from the SBOM. Accurate build and version information of components will allow users to set up automatic checks against newly released vulnerabilities. Keeping SBOM updates tied to software updates decreases the burden on suppliers as well.

I. **Risk Management:** Not all vulnerabilities in software code put operators or users at real risk from software built using those vulnerable components, as the risk could be mitigated elsewhere or deemed to be negligible. One approach to managing this might be to communicate that software is ‘‘not affected’’ by a specific vulnerability through a Vulnerability Exploitability eXchange (or ‘‘VEX’’),[14] but other solutions may exist.

LLNL is working on tools and methods for doing system-level impact analyses. One of the goals of these tools is to determine whether a vulnerability in software or firmware running on a device in a critical system causes a greater site level impact or system level impact. This is an important piece for a comprehensive risk management framework.

J. **Flexibility of implementation and potential requirements:** If there are legitimate reasons why the above elements might be difficult to adopt or use for certain technologies, industries, or communities, how might the goals and use cases described above be fulfilled through alternate means? What accommodations and alternate approaches can deliver benefits while allowing for flexibility?

Instructions for Commenters: NTIA invites comment on the full range of issues that may be presented in this Notice, including issues that are not specifically raised in the above questions. Commenters are encouraged to address any or all of the above questions. Comments that contain references to studies, research, and other empirical data that are not widely available should include copies of the referenced materials with the submitted comments. Comments submitted by email should be machine-readable and should not be copy-protected. Responders should include the name of the person or organization filing the comment, which will facilitate agency follow up for clarifications as necessary, as well as a page number on each page of their submissions. All comments received are a part of the public record and will be posted on *regulations.gov*