Billing Code: 3510-60-P

DEPARTMENT OF COMMERCE

National Telecommunications and Information Administration

Software Bill of Materials Elements and Considerations

[Docket No. 210527-0117]

RIN 0660-XC051

AGENCY: National Telecommunications and Information Administration, U.S. Department of Commerce.

ACTION: Notice, request for public comment.

SUMMARY: The Executive Order on Improving the Nation's Cybersecurity directs the Department of Commerce, in coordination with the National Telecommunications and Information Administration (NTIA), to publish the minimum elements for a Software Bill of Materials (SBOM). Through this Notice, following from the Executive Order, NTIA is requesting comments on the minimum elements for an SBOM, and what other factors should be considered in the request, production, distribution, and consumption of SBOMs.

DATES: Comments are due on or before **[INSERT DATE 15 DAYS AFTER DATE OF PUBLICATION IN THE *FEDERAL REGISTER*]**.

ADDRESSES: Written comments may be submitted on this document identified by NTIA-2021-0001 through www.regulations.gov or by email to SBOM_RFC@ntia.gov. Written comments also may be submitted by mail to the National Telecommunications and Information Administration, U.S. Department of Commerce, 1401 Constitution Avenue NW, Room 4725, Attn: Evelyn L. Remaley, Acting NTIA Administrator, Washington, DC 20230. For more detailed instructions about submitting comments, see the ‘‘Instructions for Commenters’’ section at the end of this Notice.

FOR FURTHER INFORMATION CONTACT: Allan Friedman, National Telecommunications and Information Administration, U.S. Department of Commerce, 1401 Constitution Avenue, NW, Room 4725, Washington, DC 20230; telephone: (202) 482-4281; email: afriedman@ntia.gov. Please direct media inquiries to NTIA’s Office of Public Affairs: (202) 482-7002; email: press@ntia.gov.

SUPPLEMENTARY INFORMATION:

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

[1] Exec. Order No. 14,028 of May 12, 2021, 86 Fed. Reg. 26,633 (May 17, 2021).

[2] See David J. Redl, NTIA Launches Initiative to Improve Software Component Transparency, Nat’l Telecomm. & Info. Admin. (June 6, 2018), https://www.ntia.doc.gov/blog/2018/ntia-launches-initiative-improve-software-component-transparency; Allan Friedman, Dir., Cybersecurity, Nat’l Telecomm. & Info. Admin., Transparency in the Software Supply Chain: Making SBOM a Reality, Address at Black Hat USA 2019 Conference (Aug. 7, 2019).

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 multistakeholder 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

[3] NTIA, Multistakeholder Process on Promoting Software Component Transparency, Notice of Open Meeting, 83 Fed. Reg. 26,434 (June 7, 2018).

[4] See Seth Carmody et al., Building Resilient Medical Technology Supply Chains with a Software Bill of Materials, 4 npj Digit. Med., at 1, 1 – 2 (2021), https://doi.org/10.1038/s41746-021-00403-w.

[5] See Susan Miller, Protecting the Supply Chain with a Software Bill of Materials, GCN (Feb. 22, 2021), https://gcn.com/articles/2021/02/22/sbom-supply-chain-security.aspx.

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.

Philips: Although there is value, we should recognize that use cases are limited to ONLY identifying component level vulnerabilities. Based on a SBOM one can only claim that the product/software has integrated a component with certain vulnerabilities. Whether this has any exposure beyond the product is determined by the design. SBOM is not usable for determining product level risk except for the product design owner.

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

  Philips -To be defined how to use this for open source. Currently, many variations exist:

  - Name of the component

  - URL of the repository (e.g., a github URL)

  - URL of the website owning the component (e.,g.,a link to a component on Apache.org)

  - Name of the author and probably more.

  Then, communities split, people clone repositories and these repositories get a new life, repositories move from one provider to another provider et cetera. Procedures how to deal with this will be needed.

- Component name

  Philips- Note that multiple components can exist with the same name. And that components change their name over their life cycle, especially if the marketing department is involved.

- Version of the component

Philips-It would be good to give guidance on the semantics of versioning.

As an example, with Microsoft the it is not clear what the version is of a component. On one hand, we get all kinds of names like Windows releases, editions, service packs et cetera. It is rarely clear which is a new version of another one and which is an alternative but similar component.

On the other hand, on the system, Microsoft components have an interface that you can use to query their version. This has a well-defined structure in the more conventional x.y.z form and probably is more reliable to detect changes. The version through this interface does not seem to have any relation at all with the releases, editions, service packs and whatever else the marketing people have thought up.

Clarity is needed which to use and unambiguous identification of both variants and versions should be a requirement on the providers of software components

Clarity is needed on how the supplier of the software should provide this information. The same software component may be provided through multiple suppliers. For example, the Postgress database can be obtained directly as open source, or as part of a Linux distribution with or without a commercial service contract. Would be good to understand how to use the combination of ‘supplier’ and ‘author’ fields to identify such situations. This may be the same version from the same sources with the same binaries, or different versions, changes to the sources or different binaries (for example because of compilers and compilation settings).

The same software component may be provided through multiple suppliers with different binaries because it is compiled on different environments. Vulnerabilities may also result from different compiler settings, for example, accidental inclusion of debug code which normally would be stripped out by a compiler switch.

Should SBOM track the sources, the binaries, or both?

- Cryptograph hash of the component

  Philips: This might technically be feasible but difficult to maintain for complex components, e.g. operating systems which will be different hashes after every patch cycle and it needs to be documented which files are included in the hash, etc. The perceived added value for some components is limited and a lot of effort. The hashing method used to calculate the hash should be included as part of this field or added as a separate field. Furthermore none of the public registers include.

- Any other unique identifier

- Dependency relationship

  Philips: For main components on a system-of-systems, e.g. where a separate SBOM is provided this is key. If the component could be shown in relationship to system functionality it would add more value, for relationship to other components it might have limited value for risk assessment.

- Author of the SBOM data

  Philips: Providing personal data of manufacturer employees is disproportional, the supplier name or the name of the legal manufacturer should be sufficient, also unclear why this information is required per component.

  How to disambiguate authors, how to protect privacy? Are authors persons writing the software, or the companies owning the software?

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]

[6] See generally Framing Working Grp., Nat’l Telecomm. & Info. Admin., Framing Software Component Transparency (2019), https://www.ntia.gov/files/ntia/publications/framingsbom_20191112.pdf. (providing further information on baseline components).

[7] Framing Working Group, Nat’l Telecomm. & Info. Admin., Software Identification Challenges and Guidance (2021), https://www.ntia.gov/files/ntia/publications/ntia_sbom_software_identity-2021mar30.pdf.

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.

  Philips: SBOM could also be generated on the product.

  Note that on many products there are independent update cycles of 3rd parties that necessitate updating. This is especially the case if for example foundational updates like the OS are done directly by a HealthCare customer. What should the scope of SBOM in such a case be? Supplier may do an initial delivery of operating system and will provide updates of the application. Updates of the operating system may be done by the hospital itself, so supplier does not know what will exactly be on the system. This is especially the case for non-safety critical applications like diagnostic applications, but also apps on mobiles.

- 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.

Philips: SBOM and dependencies might not be sufficient for use cases for considering and determining resulting risk. Risk determination requires deep product design knowledge from the product design owner.

- 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.

Philips: SBOM could also be generated on the product.

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]

Philips: More standardization is required to achieve automation in an enterprise, such as the discovery of devices (manufacturer/product/version) to trigger a new SBOM download as products might be continuously updated outside of the control of the IT department (e.g., medical devices are typically not managed by the hospital but by the vendor or a third party). The use of MUD files is one of the proposals being discussed but also other standards could be developed for instance to download SBOMs (in)directly from the device.

Tooling: Relation to other standards and RFCs, e.g., MUD and the ietf drafts on SBOM as part of MUD? Preferably standards based and combine with other functionality.

[8] Framing Working Grp., Nat'l Telecomm. & Info. Admin., Sharing and Exchanging SBOMs (2021), https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf.

[9] See also SPDX, https://spdx.dev/ (last visited May 18, 2021).

[10] See also CycloneDX, https://cyclonedx.org/ (last visited May 18, 2021).

[11] See David Waltermire et al., Guidelines for the Creation of Interoperable Software Identification (SWID) Tags (2016) (Nat'l Inst. of Standards & Tech. Internal Rep. 8060), http://dx.doi.org/10.6028/NIST.IR.8060 (SWID tags are defined by ISO/IEC 19770-2:2015).

[12] See, e.g., SwiftBOM – SBOM Generator for PoC and Demos, https://democert.org/sbom/ (last visited May 18, 2021).

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.

Philips General:

How to deal with large software packages that can be hardened (i.e., unnecessary components removed). This is for example a practice on Linux (part of the DoD STIGS) as well as on Windows. It is also a practice at some customers who will install additional software themselves, or who will remove some of the software themselves**.**

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?

   Philips: No, much more standardization is required, especially in the areas of naming conventions, device discovery and SBOM download (from the product or a vendor's portal) and VEX to add true value to the use of SBOM.

   For the license use case it is essential to know :

   - How is the software distributed (e.g. SaaS, Open Source Software, Commercial Distribution),
   - Is the software modified (original/ modified)

- What is the software composition? (as source code, Static/Dynamic Linking)

Due to scattered ecosystems of dependency management, build tooling etc, one practical concern is reliable retrieval and conformance for “correctness” of the data fields. *Provenance* aspects should be also considered as part of the data fields.

It is very important is that there are not only data fields but also requirements / standardization of the information provided as part of the data fields.

Some examples:

- Commercial software, Microsoft: there are no unambiguous identifications of a lot of the Microsoft software being used as part of products. There are unique codes for individual licenses but these are too fine grained, for example because they include different license model information for the same software. Then there are product names which are totally confusing, inconsistent across different Microsoft product families and changing names frequently (e.g., the Microsoft Windows LTSB which became LTSC even for the same software, the naming of SQL Server with editions, releases, the use of years as an identifying name with the year often not being the same as something was released).
- Open source software: the same name may be reused by many different parties. Parties can clone software which is in an open source repository. Even worse, the original URL to the open source repository can vanish so that only multiple clones are left. Another person can re-use the same URL in for example github to replace the previously existing component at that URL.

Standardized approaches of dealing with such situations are needed, especially if they will also be taken into account by commercial software vendors and by the people who provide important parts of the open source ecosystem like github, Atlassian, sourceforge et cetera..

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

   Philips: Complex use cases are not envisioned, such a product consisting of multiple (semi-independent) hardware/software components (system of systems) and thus perhaps producing multiple SBOMs for a single 'product'.

   Additional use case where SBoM is required is Export Control Regulations for Software. Some countries' export control regulations may require taking additional steps to ensure that an open source project is satisfying obligations under local laws. See: https://www.linuxfoundation.org/wp-content/uploads/UnderstandingOpenSourceTechnologyandUSExportControls_Whitepaper_070220-1.pdf.

   Another use case where SBoM is useful is for tracking the usage of InnerSource Internal Components. One can apply same mechanisms and practices to inner source components as for open source components (especially for security vulnerabilities)

2. 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.

Philips: It is vital to resolve this concern.

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.

Philips: Based on above statement is not transparent why we distinguish Software-as-a Service and online services, since even on premises one can apply the Zero Trust Principles.

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.

d. 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.

   Philips: This is not only the integrity and authenticity of the SBOM, are there any thoughts on the integrity of the system that claims to have a certain SBOM? Binary hashes might help here, and mechanisms as part of for example certain Linux package managers that can verify whether the software on the system is still the same as the software that should be on the system according to the package managers. People have been copying binaries from one system to another in the past, effectively invalidating the SBOM for that system in a way that is difficult to detect if there is no good information (and tooling) to check the actual system against the SBOM

e. 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?

Philips: Security best practices for the development and manufacturing sites should be applied to address this issue, SBOM has limited to no value to support this.

f. 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?

[13] Exec. Order No.14028 § 4(e)(i) – (x), 86 Fed. Reg. 26,633, 26,638 – 39 (May 12, 2021).

g. 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.

Philips: A means to provide the SBOM from the system itself is preferred as this could resolve system-of-system issues, integrated online services, products that have optional software/hardware components. This could result in a single system to provide multiple SBOMs.

h. 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.

Philips: If you use a standard docker image, you could stop at the docker level, and provide the name/version instead of the decomposition of the docker. If however the docker image is modified a full decomposition is necessary to ensure all modifications are properly known.

i. 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.

Philips: Vulnerabilities should be kept separate from the SBOM for maintenance reasons.

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.

Philips: The use of VEX is essential, without VEX the number of false perceived risks will impact the practical value of SBOMs.

3. 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

[14] David Braue, Software ‘Bill of Materials’ To Become Standard?, Info. Age (Oct. 22, 2020, 11:34 AM), https://ia.acs.org.au/article/2020/software-bill-of-materials-to-become-standard.html.

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 and the NTIA website, https://www.ntia.gov/, without change. All personal identifying information (for example, name, address) voluntarily submitted by the commenter may be publicly accessible. Do not submit confidential business information or otherwise sensitive or protected information.

Dated: May 27, 2021.

Kathy D. Smith,

Chief Counsel, National Telecommunications and Information Administration.