June 17, 2021

VIA EMAIL: SBOM_RFC@ntia.gov

Attn: Evelyn L. Remaley
Acting NTIA Administrator
National Telecommunications and Information Administration
U.S. Department of Commerce
1401 Constitution Avenue NW,
Room 4725,
Washington, DC 20230

Comments on NTIA's Request for Information (RFI) on *"Software Bill of Materials Elements and Considerations"*

The Cybersecurity Coalition ("Coalition") submits this paper in response to the National Telecommunications and Information Administration's Request for Information on "*Software Bill of Materials and Considerations*," posted on June 2, 2021.

The Coalition is composed of leading companies with a specialty in cybersecurity products and services, who are dedicated to finding and advancing consensus policy solutions that promote the development and adoption of cybersecurity technologies. We seek to ensure a robust marketplace and effective policy environment that will encourage companies of all sizes to take steps to improve cybersecurity risk management.

As we begin our comments, it is important to recognize SBOMs are not commonplace today in the vast majority of delivered software. SBOM development is very much a work in progress. There are no specifications defining the business and operational rules of an SBOM. Government agencies are discussing and considering requiring SBOMs as if they are real and exist today. In fact, the definition of SBOM found in Section 10 of the EO is not a real-world definition of on-the-market-today solutions beyond research, proof-of-concept, and demonstration projects.

> *(j) the term "Software Bill of Materials" or "SBOM" means a formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open source and commercial software components. The SBOM enumerates these components in a product. It is analogous to a list of ingredients on food packaging. An SBOM is useful to those who develop or manufacture software, those who select or purchase software, and those who operate software. Developers often use available open source and third-party software components to create a*

product; an SBOM allows the builder to make sure those components are up to date and to respond quickly to new vulnerabilities. Buyers can use an SBOM to perform vulnerability or license analysis, both of which can be used to evaluate risk in a product. Those who operate software can use SBOMs to quickly and easily determine whether they are at potential risk of a newly discovered vulnerability. A widely used, machine-readable SBOM format allows for greater benefits through automation and tool integration. The SBOMs gain greater value when collectively stored in a repository that can be easily queried by other applications and systems. Understanding the supply chain of software, obtaining an SBOM, and using it to analyze known vulnerabilities are crucial in managing risk.

The SBOM definition in this section of the EO is aspirational. That said, we believe this vision is the direction for the software industry should take, in the critical context of promoting secure software supply change management practices more broadly. We are highly supportive of SBOMs in general. There is, however, still a great deal of work required to define and document what SBOMs are and to make SBOMs prevalent within commercial software. The U.S. Government should not be considering requiring vendors to produce something before real specifications are documented, as to what is required, how they work for specific use cases, and generally understood by government, those required to provide them, and those that would use them on the customer side.

Potential Uses of an SBOM

The Coalition believes SBOMs can provide the solid foundational capability for critical future software supply chain functionality to address asset management, vulnerability, and software naming issues, at a minimum. There are many potential use cases that could benefit from software information published in an SBOM. They include:

- **Package Verification** – Verification of installation media at acquisition and at the time of installation would rely on cryptographic hashes and/or digital signatures specified on and in the SBOM file to assure all the components of the software package are installed on the disk appropriately.

- **Software Inventory** – Validation by asset tools in locating and determining the numbers and currency of the installed products. This allows a site to discover if they are running more than they should and that the version of the software being run is the current version with the latest patches.

- **Software Integrity** – Software tampering detection is not a capability that has seen widespread use in the software industry but can be critical to assuring the pieces and parts of the software package or suite/bundle have not been altered by malicious actors or by accidental disk corruption. Signed (by digital signature) SBOM files with cryptographic hashes for the individual parts that make up the software can be used to assure the software provided by the publisher is what was delivered and should execute in the manner the software publisher expects it to. This can be done just prior to execution or on-demand, validating the cryptographic hashes to assure the software and its associated components have not been tampered with.

- **Component Vulnerability** – Provide a means for correlating known vulnerabilities in third-party software and the identification of patches and updates to remediate a vulnerability. An enterprise can determine what components of their environment are vulnerable based on a

known library vulnerability (e.g., Heartbleed). Not knowing when there are vulnerable software components in the enterprise environment that have been identified as having a critical vulnerability serves no one but the attackers. It is important for network and security operations staff to be able to quickly identify the vulnerable software that needs to be updated or mitigated. Being able to see what third party components (if any) are included in software provides the ability for the user community to be able to know the true security posture of their deployed software.

It is also important to know what products are not subject to specific known vulnerabilities. Today commercial and open-source library components are commonplace and while many components have a common origin, their support and evolving development often take different paths. If a vulnerability is discovered in an open-source version, it does not mean that all versions derived from that project are vulnerable. The same holds true for derived software. If a commercially derived version includes software with a known vulnerability, it does not necessarily mean the community or upstream open-source version does. It is important to be able to accurately identify and distinguish which third party components do and do not contain known vulnerabilities as it will reduce the organizational activities needed to respond.

- **Procurement Transparency –** Construction of software may include third party libraries that are not desired by the organization purchasing it. There may be other considerations as to the development of the software, such as the percentage of third-party code to vendor code, those in control of a third-party project development, etc. Requiring third-party component information to be specified in an SBOM allows the end-user organization to be able to set their own rules for what are acceptable third-party components to run in their infrastructure.

- **Development Support** – An SBOM can assist developers in exactly knowing which version of the software they are running while developing either additional features or bug fixes. There is no ambiguity when it comes to addressing a customer problem as to which version of the software, when it was built and from what components, the bug affects. Hashes and/or digital signatures can be used to assure the developer or tester is recreating a bug with the exact version a customer has on-site. This too makes it more efficient for the developer to address external issues, whether reported as a functional problem or a security issue. If SBOMs are automatically generated for every single build (as reality requires), developers can now use this information to document development issues before the product is ever released.

- **Product Support -** SBOMs provide the ability for product support teams to quickly determine what version of their software the customer is calling about. They can ask the customer to read entries from the SBOM file or simply share it on-screen during a video call. Immediately, support staff knows what it is they're dealing with, what version and can use that info to determine if there are additional updates the customer should install or information the customer should know about. This reduces the support costs as it minimizes the amount of time that support staff needs to be on the phone with any specific customer.

These use cases could all benefit from an SBOM. **However, much work remains to develop a cohesive, practical, and widely accepted approach (much less a specific implementation). As such, we strongly urge NTIA to keep its recommended minimum requirements simple.**

Product Naming

One of the extremely hard challenges the software industry has been trying to address is around product naming. Software creators/publishers/vendors have multiple ways to name their software. They create

product names for families of software, they create individual product names, and they create different naming for marketing purposes. Sometimes version information further complicates product naming. Often a product is known in the market by three or four "common" names. Combine that with the varying ways that product versioning is done across the market, and we have a significant challenge stemming from how to definitively denote which assemblage of software is being referred to at any given date and time.

Efforts around producing consistent naming for software products have produced solid standards for syntax and usage. The assumption with those standards was that vendors would use them to name their products. It was hoped that vendors would supply "official" names for their products that industry could use and recognize. This has never happened in a universal way valuable to the marketplace. The NIST Common Platform Enumeration (CPE) was seen as the way forward at one point in hopes of solving the naming problem. What NIST and industry discovered was without software vendors supplying their official product names, we still had the same problem. It is critical that the responsibility for proper product naming rests in the hands of the vendors. Only they can designate an "official" product name.

The product name specified in SBOM files must be understood moving forward as being the "official" name for the product the SBOM file is associated with. Using SBOM files to allow the producing organization to define this for industry-wide usage is vital to properly addressing the product naming issue. If SBOMs are widely adopted, and the SBOM files are generated as an automatic part of the build process, this would allow vendors to indicate an official name without having to specify it "out-of-band" as was required by efforts such as CPE. This becomes just an artifact of the SBOM generation at build time. We believe this is the proper way to address the naming issue and do so as a natural outcome of the use of SBOMs.

Additionally, names are not permanent. Acquisitions of companies and corporate carve-outs can cause products to be renamed. It would be useful to have an optional data field for those products that have had their name officially change. An "alias" or similar field could be specified so as to allow the previous official name to be recognized for use in inventory, vulnerability or procurement use cases.

Hardly anything gets less complex

The use of software bill of materials should be focused on what a traditional Bill of Materials has been in the past. It is a description of an "as built" delivery of the package. In this case the package is software. Construction of software today quite often incorporates third-party open-source projects and its related components into the proprietary software package. Transparency is important for emerging procurement requirements and providing the means for network staff to be able to quickly identify vulnerable software that needs to be updated or mitigated. Being able to see what third party components are included in software provides the ability for the customer community to be able to know the true security posture of their deployed software.

The Depth discussion and subsequent question in the RFC violate the keep it simple approach. Tracking all dependencies for the sake of tracking all dependencies at all levels should not be included in each and every SBOM. Product SBOMs should indicate the third-party component(s) incorporated into the product. There is no real value to the producer or purchaser to clutter up an SBOM with massive amounts of

information that is not relevant to the specificity of what is a direct part of the product. It will rarely be completely possible to accurately list every single dependency at all levels for every open-source project all the time. If this is required, every SBOM will have a statement "Known unknowns" in the SBOM. Dependency information could be provided on request of the product vendor or third-party commercial or open-source project if there is a need to see all the dependencies of dependencies at all levels. Otherwise, the only thing this accomplishes is to make the SBOM files nearly unreadable and massive in size. Many commercial products use the same open-source projects. How many sets of "openssl" dependencies should one network have to be burdened with? This type of dependency information should be stored once on a network to be queried if desired. It should not be incorporated directly into every single product SBOM for every product the organization owns and uses. Additionally, requiring this type of massive dependency information in an SBOM, makes automatic SBOM creation during the build process highly problematic. This type of requirement is better suited for additional tooling, not an SBOM.

SBOMs harbor a real change for the software industry. These are not something the industry has provided with software in the past. Anytime you introduce something new that causes a great deal of change to how companies produce their products, it can take a bit for a real understanding of what is needed and what needs to change to make that change successful. For that reason, **we request NTIA to not try and solve the entire software supply chain issues in a SBOM** and confine its guidelines for the purpose of the EO to the minimal data fields suggested for the SBOM, while leaving other key questions of delivery, SBOM communication and the like, for further discussion and standardization efforts. Questions in the RFC seem to indicate NTIA staff are considering such a path. For example:

> *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?*

SBOM's are NOT a security assurance in and of themselves. They are an information source describing the software as built with some additional meta-data.

We urge NTIA to keep its approach to SBOMs simple. Trying to do too much, too quickly with potential regulations requiring immediate change is a recipe for disaster. Not all companies creating software are big companies. Many software vendors are small companies and as such are constrained by resources and staff. The costs that these changes are leveraging on the product development community could have unintended consequences. Starting in a focused manner and evolving as use cases and needs become apparent will have more of a chance for successfully changing the overall software development environment.

We strongly agree with NTIA, as stated in the RFC, that SBOMs can ride on 'existing mechanisms' and such delivery can 'reflect the nature of the software as well'.

The Coalition believes that work on defining, and documenting SBOMs in support of specific use cases must continue. There is a real need for documented practices and standards to be developed in order to properly define SBOMs. This is vital work if the overall longer-term goal is to mature secure software supply chain practices. But note, an SBOM will not solve the issue of software supply chain by itself. It is just one piece of the puzzle, a valuable one, but simply a piece. Keeping SBOM development focused on agreed to use cases will allow it to become more useful in less time.

The Coalition thanks the NTIA for allowing us to contribute our thoughts and recommendations to the dialog. As the conversation around this topic continues to evolve, we would welcome the opportunity to further serve as a resource on both technical and policy questions to ensure that SBOM incorporation into the software development industry is successful in driving consistent, effective cyber risk management practices.

Respectfully Submitted,

The Cybersecurity Coalition