Accenture International Limited
1 Grand Canal Square | Grand Canal Harbour
Dublin 2 | DO2 | Ireland

June 17, 2021

Evelyn Remaley
Acting Administrator, National Telecommunications and Information Administration
U.S. Department of Commerce
1401 Constitution Ave NW
Washington, DC 20230

Subject: Accenture Commentary on Software Bill of Materials, Elements and Considerations

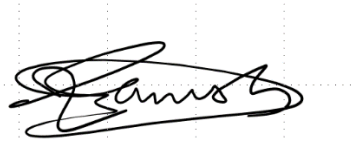
Dear Administrator Remaley,

Accenture Security appreciates the opportunity to share commentary regarding the National Telecommunications and Information Administration's (NTIA) minimal elements for a Software Bill of Materials (SBOM). We acknowledge the benefit of this undertaking and are fully supportive of the NTIA's commitment toward a more resilient and transparent future in software security.

Accenture helps businesses prepare, protect, detect, respond, and recover along all points of the security lifecycle. Leveraging our global resources and next-generation technologies, we create integrated, practical solutions that are tailored to each organization's specific business goals and industry — solutions that clients can put in place immediately. Whether defending against known threats, quickly detecting and responding to the unknown, or running an entire security operations center, we help harden organizations and make it extremely difficult for even the most sophisticated cyber adversaries to succeed.

This commentary addresses both our general thoughts and feedback on the content presented in the NTIA's request and responds to the specific questions raised. It is influenced by the feedback Accenture has received from various chief information security officers across multiple industries during working group discussions that Accenture has led to discuss the executive order, as well as from vendors in this space. We look forward to actively engaging in conversation around enhancements made to the considerations lined out in this document and are open to constructive feedback from all concerned parties.

Sincerely,



Ganesh Devarajan
Global Application Security Lead
Accenture Security

General Thoughts & Feedback

NTIA should clearly define "Cryptographic Hash". Including a cryptographic hash as a minimum requirement for identifying components will be critical for this effort to succeed. For the hashes to be useful across organizations, there must be a well-defined methodology for calculating hashes of components per programming language (Compiled code vs Scripting). For example, it is not immediately obvious how to calculate the hash of a 3rd party component in Python; if broadly instructed, a user could hash the individual file that the script is utilizing, or alternatively zip the entire 3rd party library and hash the zip file. To avoid ambiguity and promote consistency and continuity across all organizations, the NTIA should define a clear set of guidelines for software teams to generate the digital signature. Until this hash generation is standardized, it will be difficult for both the organization and those validating the dependencies to effectively match and trust the hashes.

NTIA should clearly define explicit access management protocols for the SBOM inventory and transmissions therein. Simply aggregating software components into a publicly accessible database provides malicious actors a one-stop-shop to know where to target attacks on known vulnerabilities. As soon as a new security threat is discovered and made available to the public, attackers could quickly query the inventory to understand where a high concentration of these vulnerabilities may exist, to target specific companies, governing bodies, or critical infrastructure. Additionally, such a database of SBOM's could expose organizations with a higher concentration of outdated/unpatched components, which could be indicative of a weak or immature overall security posture. Though they may be compliant and willing to share their data with NTIA and other governing bodies, organizations could be inadvertently putting themselves at risk to such targeted attacks. It goes without being said, but the inventory or application containing all of these third-party relationships and vulnerabilities should undergo regular intensive security testing to ensure strict, explicit user access and overall security compliance.

NTIA should act as a clearing house for the SBOM from all the software providers and if a new vulnerability is found in one of the commonly used 3rd party libraries then NTIA should lead the notification of the vulnerabilities privately to all the organizations impacted by that vulnerability.

NTIA should establish and publicize the "golden SBOM document", as well as a compilation/documentation guide for generally accepted programming languages or software. In an initial effort to standardize the format of the SBOM, a comprehensive and thorough example of an SBOM should be developed by the NTIA and made publicly accessible via the NTIA website. This document should be in the desired format with minimal elements clearly defined, along with examples of technologies, programming languages, and third-party libraries listed for organizations to reference. Additionally, a guide to both identify and document all known first-level (direct) dependencies should also be made available for most common programming languages, web frameworks, of software. When organizations are asked to identify and document all dependencies in an application or software, an established set of rules would help standardize the structure and comprehensiveness of the SBOM, and would eliminate any potential inconsistencies in dependency disclosure.

Responses to 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?

The potential elements described are a solid foundation to build upon, but the NTIA should consider differentiating between (A) the need for a comprehensive inventory of an organization's dependencies and third-party libraries in their code, and (B) the potential security risks when a vulnerability is discovered in a common third-party software. When the two pieces are connected the organizations are better equipped to both fix and mitigate the risk faster.

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

SBOM is a great starting point for addressing the inventory problem of how we as a nation determine which components are used by which organizations/applications. Once the inventory problem (described above) is well addressed, then the problem of connecting the inventory to publicly available vulnerability information is a logical next step. We do not envision this being disclosed to the public, but rather compiled and shared with the SBOM submitter, organization-by-organization. In other words, Company A (or Attacker A) does not have any reason to know the libraries and dependencies used in Company B, but a governing body could disclose any known vulnerabilities *in* Company A *to* Company B to aid in remediation and fostering a more secure organization.

While defining the minimum requirements and standard for SBOMs, it would benefit all participating parties to create the standard with the understanding that organizations will likely expand on it in the future. For example, if NTIA defines a JSON structure for the minimum requirements of a SBOM then it would be beneficial to include a "future work" section with recommended approaches for expanding the standard to include additional information in the future, once agreed upon by the governing body. Building in room for growth on the front-end would minimize re-work for organizations down the road, yielding quicker compliance and cooperation.

3. 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.

3a. 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.

The "source of truth" in the software component namespace should be consistent with that of the vulnerability classification namespace. In other words, the key identifiers of a vulnerability that exists in a specific piece of software should be present in that of the SBOM (namely vendor, version, and type/classification/usage of the software, and cryptographic hash), simplifying the direct mapping of vulnerability to exact software or library. The governing body may create a universal namespace by aggregating commonalities in these existing naming conventions, and

then providing a comprehensive mapping of these many existing naming conventions to the universal NTIA standard.

3b. 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 cloud-based services, there is a distributed load of security responsibilities. For the basic cloud infrastructure and foundational services, it should be the responsibility of the cloud provider to disclose and remediate underlying security issues in their libraries. These should also be standard across all users of a SAAS platform globally, unless additional dependencies are introduced by the customers themselves. If this is the case for specific services customer use, and there is not a guarantee the security is maintained by the cloud service provider, the responsibility of disclosing and maintaining the application's security compliance falls on the customer. If any additional libraries are introduced beyond those disclosed by the SAAS or service provider, the customer will be liable to provide a list of third-party tools for their individual application.

3d. 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.

Given the recent attacks injected the malicious code during the build process, there needs to be a way to recreate the compiled code outside of the software provider organization to verify the integrity of the software and also be able to recreate the SBOM in a trusted manner that can be certified and attested similar to how the PCI council does.

3e. 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?

Please see 3d. Both the compiled code and the SBOM should be able to be generated by a clearing house and that organization can further sign and attest the components for more public usage. Knowing very well that this could become the single most targeted organization in the world for attackers.

3f. 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?

No initial commentary – DevSecOps/CICD commentary requested.

3g. 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.

No commentary.

3h. 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.

For organizations with hundreds or thousands of complex applications, it would be virtually impossible to manually generate a complete and comprehensive list of nested dependencies for their software. While this process is automatable, it will be difficult for organizations to fully disclose multiple levels of dependencies. For these reasons, we recommend the NTIA to provide an automated means for organizations to recursively identify nested dependencies, given an organization provides a list of first-level or direct dependencies in an application. Ideally, each library used in a piece of software would have its own SBOM in the same inventory, which would disclose the first-level or direct dependencies that it contains. Thus, any nested or indirect dependencies in an organization's code would theoretically be disclosed in another organization's entry. Pairing this with the additional automated processes provided by the NTIA would minimize any dependency on organizations to manually identify multiple levels of dependencies, while still enabling comprehensive dependency coverage in an SBOM.

3i. 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.

No commentary beyond what is described above.

3j. 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.

While vulnerabilities may not be exploitable through an application's intended functionality, a combination of vulnerabilities can be chained together to exploit these 'unaffected' vulnerabilities. Security remediations/mitigations put in place by an application team can indeed reduce the surface area of a vulnerability but cannot be guaranteed to completely remove the risk. When remediating a cross-site scripting vulnerability, for example, a developer can hard-code a sanitization mechanism that can remove all special characters and deem it 'not affected' by the vulnerability, but variations and encodings of an attacker's commands can still exploit the vulnerability. For these reasons, we recommend NTIA not to allow organizations to classify vulnerabilities as "not affected" or "mitigated by design", as doing so would add a manual, subjective layer to the proposed process. Organizations should be made aware of all

vulnerabilities that exist in third-party dependencies regardless of their potential exploitability, rather than just those ‘agreed upon’ or acknowledged by the organization.