Jen Ellis
VP, Community & Public Affairs
Rapid7
120 Causeway Street
Suite 400
Boston
MA 02114
rapid7.com
jen_ellis@rapid7.com

National Telecommunications and Information Administration
U.S. Department of Commerce
1401 Constitution Ave., N.W.
Washington, D.C. 20230

June 17, 2021

The below details Rapid7's response to the National Telecommunications and Information Administration RFC on Software Bill of Materials (SBOM) Elements and Considerations (RIN 0660-XC051). Thank you for the opportunity to provide input and support efforts to strengthen cybersecurity and increase trust and transparency in the software supply chain.

Rapid7 is a US-based cybersecurity and data analytics firm, headquartered in Boston, with offices in Washington, D.C., Austin, TX, and around the world. Rapid7's solutions and services manage cybersecurity risk and simplify the complex, allowing security teams to work more effectively with IT and development to reduce vulnerabilities, monitor for malicious behavior, investigate and respond to attacks, and automate routine tasks. Over 8,900 customers worldwide rely on Rapid7 technology, services, and research to improve cybersecurity outcomes, protect consumers, and securely advance their organizations.

Rapid7 understands an SBOM to be a detailed and complete breakdown of the components of a delivered software product or service. This covers the origins and ongoing ownership of all first and third party elements. This enables users to then investigate vulnerability information associated with these components.

In general, Rapid7 believes SBOMs can help provide greater transparency and accountability for cybersecurity in the supply chain; however, the measure of its impact will reside in how it is deployed. Consistency, automation, interoperability, and as far as possible, reduction of complexity, will all be key to ensuring SBOMs drive value for security programs. In addition, Rapid7 believes there must be a path to maturity and that path should consider how organizations below the security poverty line may be able to start to interact with SBOMs and realize some limited value.

If poorly implemented, SBOM has the potential to create confusion for receivers and place a high burden on providers, with no real value to show for it. A core element of making SBOMs viable will be developing a comprehensive scheme for identifying and referring to software components, as referenced in more detail below. Additionally, since software is constantly being iterated, there needs to be greater clarity around expectations for updating and maintaining SBOMs. There must be a balance between keeping information up-to-date and over-burdening providers.

In the journey towards realizing the value of SBOMs, there is a great deal of complexity and many questions that need to be addressed. We urge the government to take the appropriate steps and time to find necessary, pragmatic solutions, and not hurry to roll this out to an arbitrary timeline that may end up causing more harm than good.

* * *

Q1. Are the elements described above, including data fields, operational considerations, and support for automation, sufficient? What other elements should be considered and why?
The elements are mostly sufficient as overarching goals and outcomes.

One potential critical data field that is missing is the concept of a “backport”. Operating system distributions such as Ubuntu Linux have package managers that bundle distributions for components such as “Apache HTTPD”. It is often the case when distribution is “frozen” that emerging vulnerabilities in such components are not updated to the current new build level, but the vulnerability fix is “backported” into the existing codebase (and, thus, package builds), yet the version identifier remains the same. Rather than reflect this condition in a sub-component of a baseline component data field, it would be advantageous to have it be a separate, structured baseline field to facilitate expedient determination of the vulnerable status of a given component.

It is also unclear how an SBOM should reflect packages that include modifications to the source. If the modifications are not captured, important information could be missed and software inventory may be inaccurate. However, if modifications require additional tailored SBOMs that can result in a heavy burden on vendors and a confusing amount of information for receivers. There needs to be some balance in the way this is addressed.

NTIA could mitigate some potential resource waste and increase adoption if there were explicitly defined ways of noting that a component is/was not exploitable in the way the supplier is using it. While each of the three aligned SBOM standards provide some ability to annotate an SBOM with this information, it would help if NTIA standardized the specification.

Furthermore, a software/hardware provider may choose to fix a discovered vulnerability internally in their own, maintained version of the component codebase but not issue, or may be unable to issue, an upstream patch to the source. The "Version of the component" baseline component field should provide a way for this to be documented to avoid errors in human or automation analyses that would report the component as otherwise vulnerable.

There are also some considerations that may be missing in associated details that will be addressed in the sub-paragraphs of question 3.

Q2. Are there additional use cases that can further inform the elements of SBOM?

The discussions around SBOM have, rightly, focused primarily on providing transparency to the external consumers of a software or hardware product. It could be beneficial (to support wider-spread adoption of SBOM) to include use cases where internal teams or product owners can use the SBOM information for decision making, e.g. whether to ship a software product with a known sub-component that has a known vulnerability that may not be exploitable in the way the component is presently used but would be safer if there was a delay until the next component release comes out.

While the SBOM is useful as a tool to document in-use components for a given consumer, large government departments/agencies or larger organizations may benefit if there is an example use case demonstrating a cross-functional, enterprise-wide view of dependencies, versions, and vulnerabilities.

Furthermore, documented use cases around the complementary relationship between SBOM and existing vulnerability remediation and management practices could also help support wider-spread adoption of SBOM outside of the initial, Federal mandate.

Q3. 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.***
There needs to be a way to either have a unified identity database/namespace — similar to a UL Product Id[1], FCC ID[2], etc. to ensure a common source of truth or the mandated ability to translate between formats without loss of granularity or integrity.

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.***
Organizations that leverage SaaS solutions are no less at risk of data loss, data theft, or loss of business continuity due to vulnerability exploitation than those that procure and deploy and maintain services internally. Despite the availability of modern toolchains, there are still large percentages of cloud-based services that do not take advantage of these advanced processes and technologies.

   SBOM requirements should include the need to provide a mechanism for consumers of SaaS services to query or receive published updates of the SBOM of any service they use. This would encourage the standardization and automation of releases which would have a positive impact on overall platform safety and resilience as well as transparency.

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.***
Relevant legacy and binary-only software should absolutely be covered in an SBOM. One approach could be to hold an identifier for each specific version of legacy or binary-only software that has an officially maintained SBOM that is updated via an open and transparent process whenever new sub-components or general vulnerabilities are identified. This does not conflict with the spirit or actual details of the present SBOM proposal since it would enable incremental transparency of the components of such sourced products.

[1] UL Certification Database | UL
[2] FCC ID Search | Federal Communications Commission

It is unclear what the expectations will be in situations where components are no longer supported or their providers do not provide an SBOM themselves - would the burden fall on the third party that is incorporating these elements into their software product be responsible for providing an SBOM for the components? If so, presumably, multiple third parties may create SBOMs for the same components - does this raise any concerns over conflation or lack of consistency? Some components may break down further into components, potentially creating a sort of layered up hierarchy of SBOMs; how many layers down should providers go - do they need to provide SBOMs all the way through the stack? These questions likely apply to legacy systems, but could also apply more broadly and may be somewhat addressed in question h. below.

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.***
Given the data formats involved and the potential ways to publish and consume SBOM, the use of battle-tested processes such as cryptographic digital signatures should provide ample means to validate the integrity and authenticity of a published SBOM all the way down through chained/dependent components. Given that a key baseline component field is "Cryptograph hash of the component", adding another cryptographic component that applies to the SBOM itself should not be an onerous task for suppliers and should be easy to consume by procurers.

   SBOM could also take cues from TLS certificate issuance and "valid from" and "valid to" fields, with the "valid to" field being blank until the component SBOM is superseded by the next generation. Semantic versioning of SBOMs could be used instead provided there is a central directory where "latest" and "current" component SBOM metadata could be retrieved.

   This is one of the areas where extra baseline component fields may be in-order.

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?

Given the difficulty of preventing a supply chain attack against generated SBOMs across all generation contexts, it may make more sense to assume there will be instances where sophisticated, capable attackers *do* compromise the generation or consumption processes and provide a means for notification of such an event (after detection/verification) occurring to enable quicker remediation. Discovery of these events remains a huge challenge.

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?

Given that not all SBOM consumers (i.e. organizations outside the scope of EO 14028) will want, need, or be able to utilize the extended attributes associated with this high assurance use case, this is an area where the baseline "required" fields can — in a fairly straightforward fashion — be augmented with a voluntary option field or set of fields to accommodate use cases where higher assurance is desired. Many higher assurance situations already have methods and procedures in place for documenting and exchanging this type of information and it should likely not be a high priority for the working group.

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.

Imposing restrictions on discovery and consumption mechanisms could do more harm than good. As an example, if the "SOAP" XML standard had been a hard — and cumbersome to amend — mandate, advancements such as JSON, REST APIs, and Graphql would likely not have been as quick to emerge. The current use of SBOM in industry is small compared to the proliferation of API-based services and SBOM discovery can only benefit by existing in the marketplace at-large and allowing innovation to surface the "best" ways to handle publishing, discovery, and consumption.

It would be beneficial if NTIA define some accepted methods to make it easier for SBOM initial adoption and provide fodder that industry groups can use to extend/enhance/refine.

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.***
This is another case where an additional baseline field for "completeness" may be in order to help organizations identify continued "unknown unknowns". However, the risk associated with opaque elements in an SBOM tree should be addressed in some way. One potential avenue could be to require providers to flag this potential risk to receivers. Another could be to apply limits for how many or how long these blind spots can be included, with associated "penalties" levied by agencies (as it pertains to the EO) and organizations (as they voluntarily participate in adopting SBOM practices).

The working group facilitating the specifics of SBOM for the EO should be tasked with defining a realistic exception process with achievable time-frames for "remediation" of the SBOM record.

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.***
Having a unique identifier (See 3a) would enable all third party vulnerability databases to provide a mapping so no direct vulnerability data needs to be included within the SBOM itself. This would also enable the referencing of vulnerabilities that have no associated CVE identifiers which is vital since many vendors now issue advisories for some vulnerability issues instead of CVE identifiers.

We reiterate the need within SBOM to identify that a given vulnerability or set of vulnerabilities is not exploitable in the configuration used in/by the component the SBOM is documenting.

j. ***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.***
The VEX concept has significant merit and is likely the path of least friction, especially given the alignment to OASIS CSAF and existing work with the various SBOM Healthcare PoCs.

Q4. 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?

Given that the current objective is to support the timely implementation of the EO, the elements described in the RFI should be considered reasonable expectations for any supplier to the Federal government. However, one accommodation that may enable faster creation and publication of SBOMs (for smaller suppliers) would be the facilitation of the creation of easily deployable open source projects that extend existing open source projects and make it as easy as possible for a supplier of limited capacity to continue to deliver products and services to the U.S. government.

* * *

Thank you for giving us the opportunity to share our views. For any additional questions or feedback, please contact Jen Ellis at jen_ellis@rapid7.com.