Dear NTIA colleagues,

BSI would like to thank you for the opportunity provided to the community to provide comments for this important topic. Please find our comments below.

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

As a starting point, the suggested data fields seem to be sufficient. However, we suggest considering the benefit of linking SBOMs as described in comment regards 3a.

Concerning the support for automation, it is important to not only suggest formats for SBOM (as e.g. SPDX) but also provide suggestions for distribution. If the distribution question is not tackled and no guidance is given, we will soon have all sorts of different ways for distributing SBOMs. This is the current situation for security advisories and we see that this is a massive obstacle against automation. Therefore, we suggest specifying three ways of distributing SBOMs:

- Online via API (e.g. with a configurable)
- Online as file based approach / repository
- Offline from the device in question

All these three way must use common open source standards.

3. 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.

The name must be unique per organization. The organization then provides the namespace for the software identity. Links between versions of SBOM should be possible so that a new SBOM links to its predecessor. It is to be considered whether it is possible to also link old SBOMs to their successors. This will allow consumers to keep track of product versions through SBOMs. It could also solve a part of the naming problems which come with mergers, sell-offs and acquisition. As products might be renamed or available under a different namespace these links will help to always find the correct SBOM. Obviously, this works only for the online (or an out-of-band) distribution as SBOMs need to be updated once a new version is available.

3 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.

Wherever possible SBOM should be generated from source and during the build process. If only the object code was available for SBOM generation, this should be clearly stated in the SBOM. It also should add a flag that unknown components might be in the product. This is important as code might have been copied, or being inserted in a way not identifiable during the SBOM generation process.

3 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.

As the integrity and authenticity is important, SBOM should be secured with existing measures. It must be ensured that namespace injection and signature wrapping attacks are impossible. To reach this goal, the complete file should be secured.

3 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.

As commented with regard to question 1: There should be only three ways of distributing SBOMs. Each of them should use at max two existing standards to fulfill the goal. A supplier must support at least one of them. Concerning the "*./well-known/sbom*" directory we suggest looking into ROLIE (cf. RFC 8322) as a standard.

3 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.

An enforced depth of 1 – listing all components of the assembled software – seems to be a good starting point. This will enable people to start generating SBOMs fast for their products without having too much trouble finding the dependencies for their suppliers. Although it is desirable to have deeper insights, we think that this is currently not possible for most of the producers. Therefore, such a requirement will likely hinder the adoption of SBOMs. Nevertheless, we recommend pushing the suppliers to a depth of 2 as this ensures that the supplier is aware of which components he added into his product.

3 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.

As data about vulnerabilities is changing over time, BSI strongly advises against including this in any SBOM. There are well-established formats to convey such information. Adding this information to SBOMs would also contradict the proposed generation frequency ("Bundled with every product version and archived by the supplier") as the SBOM would have to be updated whenever new vulnerability information becomes available. Moreover, it also contradicts the initial purpose of SBOMs to deliver a "list of ingredients" and therefore the Principle of Least Surprise (POLS). Furthermore, it would make the SBOM standard more complex and violate the DOTADIW (Do One Thing And Do It Well) principle. Including vulnerability data would also lead to a state where SBOM becomes a competing standard to the well-established formats. This altogether makes including vulnerability data to a substantial risk to the adoption of SBOM.

3 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"), but other solutions may exist.

Risk can only be assessed if sufficient information is provided. As the VEX format conveys information which can be seen as "negative security advisory" it should make use of the Common Security Advisory Framework - a standard developed by the CSAF TC at OASIS Open. This will leverage synergies as many companies already have a process to deal with security advisories. Information they provide is used to make a risk based decision with regard to vulnerabilities.