PUBLIC SUBMISSION

As of: June 19, 2021
Received: June 15, 2021
Status: Pending_Post
Tracking No. kpy-iap2-w4e7
Comments Due: June 17, 2021
Submission Type: Web

Docket: NTIA-2021-0001
Software Bill of Materials Elements and Considerations

Comment On: NTIA-2021-0001-0001
Software Bill of Materials Elements and Considerations

Document: NTIA-2021-0001-DRAFT-0002
Comment on FR Doc # 2021-11592

Submitter Information

Email: nurmi@anchore.com
Organization: Anchore

General Comment

Comment 1: Addition of stable identifiers (vulnerability data specific) to minimum SBOM element requirements.

In reference to RFC Question 1, and this section from the document:

“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.”

To facilitate the ability for operational tools/techniques to use SBOM records to look up existing vulnerabilities from canonical sources (e.g. NVD), we believe consideration should be made in SBOM formats for how vulnerability records (e.g., NVD CVEs) reference affected software. The ability to obtain or derive a stable software element identifier will facilitate the important task of ‘given an SBOM, what are the known vulnerabilities against each element in the SBOM’ accurate and deterministic. Without such an identifier, tools are independently responsible for generating an identifier that can be used when mapping detected software artifacts to vulnerability records, which creates variability in results and introduces potential for matching errors across tools.

For example, NVD CVE records (https://nvd.nist.gov/vuln/data-feeds) use CPE strings (https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe), and an associated official CPE dictionary (https://nvd.nist.gov/products/cpe) which defines the set of all CPEs that can be referenced by CVEs. With the ability to obtain or derive a stable CPE identifier for a given SBOM element, vulnerability matching against NVD CVE data will be accurate and should generate agreement of results across tools.

Comment 2: Component identifier registry

In reference to RFC Question 1, 3:

Related to the comment on stable identifiers above - we believe there should be consideration made on the value of providing a way for software suppliers to register their org names, software component names, and other required SBOM element values in a global registry, to facilitate the ability for operational and automated tools to use the information for higher level purposes (version comparisons, origin validation, vulnerability matching). The CPE dictionary from NVD (https://nvd.nist.gov/products/cpe) is an existing example of such a registry (though limited in scope, where a new CPE is only registered the first time there is a need to register a vulnerability), where an official CPE must be in place before a CVE can reference an affected piece of software. Existence of such a registry and participation from software suppliers would not necessarily be required for all elements of an SBOM, but would allow operational and automated tools to create categories of coverage (and associated risk assessments) from an SBOM (toward the identification of 'known unknowns' as referenced in the document).

Comment 3: Support for Stable SBOM Element Version Comparison

In reference to RFC Question 3(a):

In reference to the operational aspects of tools/techniques that are able to process SBOMs for higher level functions, we would suggest considering the addition of metadata that describes, for a given SBOM element's 'version' field, how version strings can be programmatically compared. While some software/packaging ecosystems have well defined methods for version comparisons, other prominent ecosystems have not, leading to the complication where automated version comparisons can be deterministically performed without further context tied to the individual software element. This is important, as functions such as 'SBOM contains packageX versionY, and a vulnerability has packageX <= versionZ as affected. Unless there is a way to programmatically determine how to compare versionY with versionZ, different tools may produce different results or be unable to make a determination entirely. Some modern vulnerability data formats have already begun to add such context (https://github.com/golang/vulndb), but we believe considering inclusion in the SBOM itself would unlock other SBOM use cases at the source of truth, namely any use case that requires the ability to compare two SBOM records with the same software name identifier, at different versions.

See for example a vulnerability report for a specific vuln/package, where the 'type' of the version is set to a known type 'SEMVER' which allows for a stable reference for how to compare version strings of this type - https://storage.googleapis.com/go-vulndb/github.com/gorilla/websocket.json.