Steve Springett and Patrick Dwyer

CycloneDX Core Working Group

Response to document NTIA-2021-0001

The CycloneDX Core Working Group is responding to NTIA's request for a position paper on the minimum elements of a Software Bill of Materials (SBOM). Recommendations are categorized by document and entry, where document is the SBOM file or payload itself, and entry is the individual components or services within the SBOM.

Recommended SBOM document elements

Author

The author may be a person or an organization, which may or may not be the supplier. In the event questions about the SBOM arise, the author should contain contact information.

Creation Tools

The tools used to create the SBOM should be present. This will allow consumers to evaluate the methods used to produce the SBOM and will provide context about the inventory within the SBOM. For example, build-time tools and binary SCA tools can both produce SBOMs, but the quality and type of data available may vary depending on the tools used. This additional context is vital when analyzing SBOM contents for various forms of risk.

Supplier name

The supplier of the software the SBOM describes should be present.

Timestamp

The timestamp the SBOM was created or last modified should be present in the SBOM.

Unique Identifier

Every SBOM should include a unique identifier, preferably a URI. The unique identifier will allow SBOMs to be linked to each other or referenced from other systems and document formats.

Version

Every SBOM should include a version number, for the SBOM itself, represented as an incrementing integer. Typically SBOMs should not usually change after software has been built. Although there can be some situations, such as manually authored SBOMs for legacy software, in which information is unintentionally omitted. Or inaccurate information is included. When a new version of an SBOM is created it should retain the existing *Unique Identifier* and increment the version.

Component

The top level component, or assembled software/device, which the SBOM describes, should be mandatory. The information required for the top level component should be the same as the *Recommended SBOM entry elements* specified below.

Security advisories

Knowledge of 3rd party components, used in an assembled piece of software or device, can enable operators to quickly ascertain if they may be affected by a particular vulnerability.

Unfortunately, this can also lead to a high number of false positives and noise. Not all vulnerabilities are exploitable. In particular, vulnerabilities in 3rd party components are typically not exploitable in assembled software or devices. Often, especially for commercial devices and software, the only party who can accurately assess the exploitability and potential severity in context of the system is the builder of that system.

In order to be able to adopt and operationalise SBOMs at scale it is imperative to have timely access to security advisories that consider the context of the system in question.

The location of a security advisory feed should be required in SBOMs for all commercially supplied software. With preference given to machine readable security advisory formats like CSAF.

Without easy access to this information, software and device operators will spend an unmaintainable amount of time on false positives.

Security advisory feeds also enable suppliers to deliver augmented security advisories for true positives. A seemingly minor vulnerability in a 3rd party component may have a significant impact and risk in the context of the system or device. Especially for systems with safety implications. Or, a critical vulnerability might have other mitigating factors in the context of the system or device.

For suppliers, the impact of downstream consumers not having easy access to security advisories, will be magnified by the number of downstream consumers performing basic SBOM analysis for vulnerabilities in 3rd party components.

CycloneDX

Recommended SBOM entry elements

Component name

Every component in the SBOM must have a name.

Version string

Every component in the SBOM must have a version. In the event a component does not have a version (e.g. an unknown file on the filesystem), then semver with shorthash should be used. For example, 0.0.0-7af186bd

Component hash

All components should include a hash. What the hash actually represents can vary based on the component being described. In the case of a package it will typically refer to the hash of the package file as available from the upstream supplier. Or in the case of an individual file it will typically refer to the hash of the on disk file.

Every component should include hashes using the algorithms as natively used by the relevant ecosystem. This supports automated comparisons between supplied components, components as described in an SBOM, and components as available from the upstream supplier. (Noting the *Pedigree and Remediation* section below.)

Some ecosystems no longer support SHA and have adopted Blake2b or Blake3. In order to validate integrity of the widest array of components, support of modern algorithms that extend beyond those approved by NIST should be required.

Unique Identifier

Every component in the SBOM should have a unique identifier. This identifier is different from the unique identifier of the SBOM itself. When used together, deep linking is possible. For example, CSAF could reference a specific component in a specific SBOM when describing advisories. (Refer to *Security advisories* section for relevance of this use case.)

Component Identity

Every component in the SBOM should include one of Package URL, SWID, or CPE as relevant to the type of component. Every component derived from a package manager or available in a repository should include the Package URL. This provides the ability to detect outdated versions, known vulnerabilities, and provides provenance. All other components should include either a SWID tag or CPE. Limitations of the NVD should not be considered. Package URL is the most widely used identifier for open source components.

Software License

Every 3rd party component should include a standard license representation. Many copyleft licenses, and especially the AGPL, may have a profound effect on security if license obligations require the disclosure of source code that would otherwise not be made public. Some licenses trigger obligations simply by calling an application over a network, or how an organisation uses the software in question. Consumers have a right to know if they will be impacted by copyleft or other license obligations that could have an adverse impact on security.

Relationships

Every component in the SBOM should include relationships to other components. Relationships supported should consist of “includes” and “depends”, both of which have security implications. Components which include other components should be specifically identified. Components that depend on other components should also be identified. In CycloneDX terms, this is referred to as component assemblies and component dependencies. Assembly use cases will provide insight into components that may have been statically linked or otherwise embedded in other components. Dependencies provide a way to identify how a vulnerable component may have been introduced and the effort involved to remediate. If components are modified from their original form, pedigree relationships (ancestor, descendent, or variant) should be used.

Services

Vendors often document the hostnames, IP addresses, and ports of services that their software requires access to so their customers can correctly configure their proxy and firewall. Unfortunately this is typically not provided in a standard machine readable format.

These external services are used to enhance functionality or to provide core capabilities to the application. Third-party software, whether a component or service, can introduce security risk.

Therefore, SBOMs should include the services the application relies on in its default state. Many applications have configurable parameters that allow the application to call out to services that are user defined. These user-defined services are out-of-scope. The focus must be on the services that are enabled by the application as delivered to the consumer. Services should include endpoint URLs, data classifications, and directional flow of data. This information can be used for the automatic generation of data-flow diagrams, feed automated dynamic security testing tools, and network monitoring use cases.

Composition completeness

The completeness of inventory and relationships should be included in SBOMs.

For software with modern software development practices, generating a complete and accurate SBOM can be trivial. However, for some software development ecosystems, and legacy systems, a full and accurate SBOM might not be practical.

As a minimum, all top level dependencies and includes should be required.

For software and devices where a full and accurate SBOM cannot be supplied, an organization should make a risk based assessment of whether or not that is acceptable.

Composition completeness is required information to enable operators to make this assessment.

Pedigree and Remediation

In an ideal world, commercial suppliers would contribute all changes upstream for modifications they make to open source components. And those changes would be accepted and merged upstream. Unfortunately this isn’t always possible.

The reason for making changes can be varied. They might be bug fixes, security fixes, or even enhancements. They might not be able to contribute the changes upstream due to organization policies, changes might be rejected by upstream maintainers, or they may have taken a critical dependency on an unmaintained component version that they can’t immediately upgrade.

When the supplier of a device, or assembled software, is shipping modified components, it is important to be able to communicate the context of any component modifications to downstream consumers.

Without this information, any modified component will appear to be tampered with when compared to upstream. Or, when a change has been made to remediate a known vulnerability, the component can appear to be vulnerable if the remediation information isn’t readily available.

As SBOM adoption increases, absence of this information, will result in a lot of manual work for device and software operators. The impact of this manual work will be amplified and multiplied for upstream suppliers. And will severely undermine the ability to operationalize SBOM production and consumption at scale across industry.