Software Identity Discussion and Guidance

Draft version 0.2 – July 6, 2020

This document is available at:
https://docs.google.com/document/d/1tOtO90AIrsHSIPvcVhvSBmkuS1mkFWIYWFNIPnjSzhQ

Introduction

This white paper was drafted to capture the challenges of software identity or "naming" as they pertain to SBOM generation and use, and offer some guidance to the broader software community on how to address these challenges.

Software identity is important and hard

Currently, there are no global authoritative sources to obtain the values for the Component Name in SBOM data. [e.g. clear, obvious example or two?] As such, two actors who compile SBOM might use two different values for the same component. Lack of clarity about a component identity can also make it difficult to map an SBOM component to vulnerability data, license data, or other data an SBOM consumer might care about.

In the first multi-organization SBOM proof of concept exercise in 2019, the lack of common identifiers was noted as a key obstacle for automation. In the perfect world, the SBOM of a given component would be the source of the "correct name." If an SBOM is to help organizations track their third party components, we need some path towards convergence of identifiers, such that two components can be identified as being the same component or different components in largely automated systems. A single, universally used set of identifiers may not be possible, especially in the short run, absent a single or central naming authority.

The sources of the current heterogenous namespace are driven by several factors, beyond the lack of a single canonical namespace. The creators of software components define names according to their own needs. Other tools have had to develop their own internal approaches. Several standards have emerged over the years, including Common Platform Enumeration, SWID tags, Package URLs, and Software Heritage ID. As organizations start to keep their own records, or integrate this data into their build processes, they might use some of these standards, or define their own internal schema. Component names cannot even be assumed to be static, as dynamics like corporate acquisitions and project forking may lead to a change in a preferred name. Moreover, the challenges of defining a namespace and the identity of digital artifacts themselves is a known hard problem.

A software component is identified by three attributes laid out in the Framing document: Component Name, Component Version, and Supplier name. The issue of component name and identity is closely related to the challenge of identifying the component supplier. In many cases, if a supplier can effectively determine the namespace of the software they write, the supplier can effectively resolve any confusion about naming. A supplier is not just the source of the code, but also has the key functional role of responsibility, including vulnerability reporting, version maintenance, and license information. However, the supplier name may not be clear cut either. For example, a commercial software company could be referred to by both Acme and Acme_Software. Project reorganization, corporate acquisition and open-sourcing of commercial packages also further complicate long term supplier identifiers.

Landscape of the challenge – Use cases

SBOM component names may be generated or even corrected in several different conditions. While a full set of use cases is outside the scope of this document,[1] it is important to acknowledge some of the more common use cases and the actors with key roles. The terms used here reflect the definitions in the document Framing Software Transparency[2] unless otherwise noted.

Base case: Supplier generation

A supplier creates an SBOM for a primary component. The supplier is the creator of the component. It defines its own supplier identifier and component name as author of the component data. The supplier is the source of truth for the baseline elements and any other information associated with the primary component. If the entire ecosystem were to adopt this approach, recursive use of SBOM data would make SBOM assembly relatively straightforward.

Secondary Authorship: SBOM stakeholder generation

In cases where the supplier has not created an SBOM, or there is too much uncertainty about the component data, an SBOM stakeholder may author the component data. The stakeholder

[1] For a more complete discussion, see the Roles and Benefits for SBOM across the Supply Chain https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf

[2] https://www.ntia.doc.gov/files/ntia/publications/ntia_naming_use_cases_-_framing_2020-04-11.pdf

may be a user of SBOM data, such as an enterprise customer, or providing key data for its users, such as an SCA tool. The SBOM Stakeholder will make a best-effort approach to establishing the baseline information. The SBOM Stakeholder will follow the SBOM available best practices on naming, and seek to replicate at least one other extant identity for the component. Based on the author field, the SBOM Consumer will be able to determine that the SBOM was not created by the Supplier of the component.

SBOM assembly: recursion

An assembling supplier using a third party component includes SBOM data about their components in their SBOM. They use component data from the primary supplier (preferred) or from another author when the primary information is not availableIf component SBOM is not available, a best effort approach should be used to establish component information. The supplier name and component name should follow best practices. If no component information is available, incompleteness of the dependency data should be noted in the SBOM.

Potential Guidance for Component Identifier

We propose the following set of guidance with the goal of driving convergence towards a smaller set of identifiers for software components in SBOMs. This model follows a basic approach of building on existing practices and organizational structures with minimal adoption of new technology or changing practices. This guidance is particularly aimed at the secondary authorship use case above, where an SBOM stakeholder must determine the appropriate data for component fields in an SBOM they create.

It follows a two-step approach. The first step is to use an existing, trusted local set of information if possible, enabling a federated approach that reflects the supplier of the software as the optimal source of information. Failing that, we suggest that an SBOM author use one of a small set of established naming practices.

Preferred Case – Existing Supplier or Coordinate Namespace

If there exists an established, well-defined namespace, the component data author should use that. This would include established package managers with unique internal names of components. [examples?] This would also include commercial suppliers that clearly communicate the identity of their components. [This would also include github repos.???] Suppliers are encouraged to use one of the widely accepted identifier schema below. However, we note that some suppliers have their own long-established naming conventions.

Alternate Case – Use an established software identity standard

If an existing, widely accepted name does not exist, or is not available through a canonical source such as a widely used package manager, or there are multiple widely-used identifiers for a given piece of software, then using an existing namespace will not be sufficient to help identify a component. While we cannot propose a path for a single name, the best practice is to 1) use an established, standardized naming schema that is 2) used by some reasonable subset of the existing community.

Condition 1) above helps the community move towards greater automation by using existing data formats and predictability. Condition 2) requires a bit more work by the component author, but moves us closer towards convergence of a name space.

This community recommends one of the below naming schema:

- SWID tags[3]
- Package URL (PURLs)[4]
- Software Heritage IDs[5]

Supplier-generated component names should, if possible, also follow one of these standards.

We recommend the schema above over the use of CPE. NIST has publicly indicated that they intend to phase out use of CPE in the NVD in favor of SWID tags.

Supplier identification

Identifying suppliers is a similar problem to identifying software components. Also, identifying suppliers may be an important part of the solution to identifying software components, as namespaces could be defined by globally unique supplier identities.

Notes on Supplier Identifiers for Global SBOM
https://docs.google.com/document/d/1uPqhi1oHxmzBF5LVxC3RDyS6JYSwVJXzjgrD2QkEGLg

[3] Note that we refer to SWID tags here in their role as a software identifier, not their broader use as a standard to convey the SBOM dependency graph data. See more here: https://csrc.nist.gov/projects/software-identification-swid/guidelines (update with more NIST guidance when available)

[4] https://github.com/package-url/purl-spec

[5] https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html

Rough draft of Supplier Registrar concept
https://docs.google.com/document/d/1z5ghyHAySCA09cj528XS_ENmHvXI9CppF2vXMtzU_60