Software Identification Challenges and Guidance

NTIA Multistakeholder Process on Software Component Transparency
Framing Working Group
2021-03-30

Ruben Ramirez
https://unsplash.com/photos/W5hUOxgB08Y

Table of Contents

1 Executive Summary

Perhaps the biggest single challenge to supply chain transparency and the Software Bill of Materials (SBOM) model is identifying software components with sufficient discoverability and uniqueness. Component identification is fundamental to SBOM and needs to scale globally across diverse software ecosystems, sectors, and markets. This paper offers guidance to functionally identify software components in the short term and converge multiple existing identification systems in the near future. The guidance can be applied today, and can be summarized as follows:

1. Preferred Case: Use Existing Identification
   Use authoritative component identification and a corresponding identification system from component suppliers.
2. Alternative Case: Select an Identification System
   If authoritative identification is not available, create best-effort identification.
   a. Use an existing component identification system, particularly one that aligns with your software development ecosystem.[1]
   b. Lacking a clear choice of identification system, select a widely-used, active, and open system (see Component Identification Systems).

An important goal of this guidance is to limit the proliferation of duplicate and contradictory component identification. To that end, always seek existing component identification and use existing identification systems when possible.

While further testing and iteration are needed, experience with existing software identification and dependency tracking systems suggests that global-scale identification will likely leverage aspects of the Domain Name System (DNS) and intrinsic identity (i.e., cryptographic hashes of software components).

This document represents the status of in-progress work and assumes some familiarity with the output of the NTIA Software Component Transparency multistakeholder process.[2] For additional details on baseline attributes, terminology, and general SBOM background, see Framing Software Transparency.[3]

[1] For example, the pip package manager and Python Package Index (PyPI) make up a component identification system for the Python development ecosystem.
[2] A range of resources on SBOM and software component transparency are available at https://www.ntia.gov/sbom.
[3] https://www.ntia.doc.gov/files/ntia/publications/ntia_naming_use_cases_-_framing_2020-04-11.pdf.

2 Global Software Component Identification

> "There are only two hard things in Computer Science: cache invalidation and naming things."
>
> — attributed to Phil Karlton, circa 1996[4]

As of early 2021, there exists no single globally authoritative source for SBOM component identification. As such, two different SBOM authors could use two different identifiers for the same component. The converse problem can also occur if two authors use the same identifier for different components. Lack of clarity about component identity can also make it difficult to map an SBOM component to vulnerabilities, licenses, or other data of interest to an SBOM consumer. In the first multi-organization SBOM proof of concept exercise in 2019, the lack of common identifiers was noted as a key obstacle to automation.[5] In a vision of future SBOM perfection, the SBOM of a given component would provide the authoritative, sufficiently-globally-unique "correct" identity of that component, authored by the originating supplier of the component.

Reasons for the current heterogeneity in component identification extend beyond the lack of a single canonical source. Suppliers of software components define names and identification according to their own needs. Several standards have emerged over the years, including Common Platform Enumeration (CPE), software identification (SWID) tags, Package URLs (purls), and SoftWare Heritage persistent IDentifiers (SWHIDs). As suppliers keep their own records and integrate this data into their development processes, the suppliers might use some of these standards or define their own internal schema.[6] Component identification assigned by the originating supplier cannot be assumed to be static either, as dynamics like corporate acquisitions and project forking often lead to changes in component identification. Even identifying and managing digital artifacts within a defined scope or namespace can be challenging.

2.1 Name, Namespace, Identification

Global identification of software components needs to be sufficiently unambiguous so that conflicts, duplicates, and other failures are rare and can be resolved. Naming is part of, but not equivalent to, identification. A name is one of many attributes that can help to identify a component. A name alone may or may not be sufficient to identify a component, and a component can have more than one name. A comprehensive component identification system will need to account for multiple names for the same component, likely through aliases or equivalency relationships.

---

[4] https://martinfowler.com/bliki/TwoHardThings.html.

[5] https://www.ntia.gov/files/ntia/publications/ntia_sbom_healthcare_poc_report_2019_1001.pdf.

[6] In support of convergence, please try not to create new identification systems, see Guidance for Component Identification below.

A namespace is a way to partition names or identifiers. Component names within a namespace must be unique. DNS is perhaps the most well-known and widely-used example of a large-scale hierarchical namespace.[7] DNS and its associated mechanisms (like Uniform Resource Identifiers, URIs, and Uniform Resource Locators, URLs)[8] could be used as the foundation for global software component (and supplier) identification. A number of existing and developing SBOM formats and component identification systems use mechanisms based on DNS.

It is unrealistic at present to expect a single global namespace for all software components. As demonstrated in other large scale identification systems (e.g., DNS and XML namespaces), a scalable solution will likely require a way to federate and interrelate different hierarchical namespaces. Furthermore, there exist identification systems that do not rely on namespaces or other types of hierarchies. For example, the cryptographic hash of a component is a unique, intrinsic identifier that is included in the Framing Software Transparency baseline attributes and is supported by existing identification systems and SBOM formats.

3 Need for Component Identification

SBOM component identity information may be generated and changed under several different conditions. While a full set of use cases is outside the scope of this document,[9] it is important to acknowledge some of the more common use cases and key roles.

3.1 Primary SBOM Authorship

In a vision of future SBOM perfection, all suppliers author SBOMs for their primary components (i.e., components that the suppliers assemble and create themselves) and define the component identifiers (including names). Suppliers are the presumptive sources of truth for the baseline attributes and any other information associated with the suppliers' primary components. When suppliers modify or fork upstream components, the suppliers assume SBOM responsibility for the new components. When suppliers transfer components downstream to users or customers, the suppliers provide assembled SBOMs. These consist of SBOMs for the suppliers' primary components and SBOMs for included upstream components. With a critical mass of participating suppliers, this recursive provisioning of SBOM data would make SBOM collection and assembly relatively straightforward.

3.2 Secondary SBOM Authorship

In cases where the supplier of a component has not created an SBOM, or there is sufficient uncertainty about the quality of the SBOM information, another stakeholder can author an SBOM. This secondary SBOM author may be an end user or a downstream supplier. The

[7] https://www.icann.org/resources/pages/unique-authoritative-root-2012-02-25-en.
[8] https://tools.ietf.org/html/rfc3986#section-1.1.3.
[9] For a more complete discussion, see *Roles and Benefits for SBOM across the Supply Chain* https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf.

SBOM data may be used to assemble other SBOMs or integrated into services such as software composition analysis (SCA) or software asset management (SAM). The secondary author should take a best-effort approach to generate the baseline attributes and follow the Guidance for Component Identification below. By observing the difference between the author and supplier names, an SBOM consumer can determine that the supplier of the primary component did not create the SBOM.

3.3 SBOM Assembly

Suppliers should, as described in Primary SBOM Authorship, create SBOMs for their primary, first-party components, obtain SBOMs for upstream components from the relevant suppliers, and provide assembled SBOMs to their downstream users. If authoritative upstream SBOMs are not available, primary suppliers should obtain SBOMs from alternative sources or create "best-effort" SBOMs. When creating secondary SBOMs, component (and supplier) identification should follow the Guidance for Component Identification. To the extent that SBOM information is secondary, missing, incomplete, or not authoritative, this knowledge (or lack thereof) should be conveyed through the SBOM.[10] Practical SBOM assembly depends significantly on the capability to share and exchange SBOMs.[11]

4 Guidance for Component Identification

A goal of the following guidance is to harmonize and reduce the number of different identification systems for software components. This guidance builds on existing practices and organizational structures with minimal adoption of new technology or changes to existing practices. This guidance is particularly intended for the Secondary SBOM Authorship use case, where the author who creates the SBOM is not the component supplier.

This guidance follows a two-part test.

1. In the first, preferred case, if an authoritative source of component identification information exists (ideally, the component supplier), then SBOM authors should use that source and identification system. This enables a federated approach that treats the supplier of the software as the primary and authoritative source of SBOM information.
2. In the alternative case, SBOM authors should select a widely-used, active, and open identification system, taking into consideration the author's software ecosystem.

This two-part test does not solve every aspect of the identification problem. For instance, SBOM formats typically provide a way to record component identity, but the formats themselves may not specify a namespace, naming convention, or an identification system.

[10] See section 2.4.1 of https://www.ntia.gov/files/ntia/publications/framingsbom_20191112.pdf.
[11] https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf.

4.1 Preferred Case: Use Existing Identification

If established, well-defined identities exist for the components in question, then SBOM authors should use those identities and the corresponding identification system. Such systems include package managers with unique identifiers for components within their scope. This would also include suppliers of proprietary software that clearly communicate the identity of their components.

Examples of existing identification mechanisms include:

- npm (package manager for Node.js);
- Java Language Specification (JLS);[12]
- Mechanisms that incorporate DNS, such as a URL to a tagged release on a code hosting platform (e.g., GitHub, GitLab, BitBucket, SourceForge, and many others).

Lacking established component identity and an identification system, suppliers and authors should use one of the widely-accepted Component Identification Systems listed below.

4.2 Alternative Case: Select an Identification System

If established, authoritative identities and an identification system do not exist for given components, then SBOM authors should select an identification system that is under active development and that matches the software ecosystem being used.

Using an existing identification system helps the community converge towards greater scalability and automation by using existing data formats and predictable tools and processes. Secondary SBOM authors should avoid creating new component identifiers or identification systems if at all possible.

4.2.1 Component Identification Systems

While a universal, global component identification system remains a challenge, practical, smaller scale options exist. The following component identification systems have been considered:

1. package URL (purl);[13]
2. SWID tags;[14]
3. Software Heritage IDs (SWHID)[15].

---

[12] https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html.

[13] "... package URL is an attempt to standardize existing approaches to reliably identify and locate software packages" and works with many different identification systems, see https://github.com/package-url/purl-spec.

[14] SWID tags are mentioned here for their capability as a software identifier, not their broader use as a standard that can convey SBOM dependency relationships. See https://csrc.nist.gov/projects/software-identification-swid/guidelines.

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

Component identification is a required part of a comprehensive SBOM solution, but additional data, formats, and tooling are needed.[16]

As noted previously, many software identification systems are based on DNS and related mechanisms like URIs or URLs. Some package managers can automatically download software given a URL, and where this URL is stable and unique to a specific version, a URL can serve as a component ID. URLs can be used to re-acquire the component and do not require a central registry beyond the use of DNS. Even if the component referenced by the URL is no longer available, the URL may still be functional for identification purposes.

Some component identification systems integrate the location or source of components. These sources may change over time or even disappear. For example, some projects migrated away from GitHub after its acquisition in 2018,[17] and 250,000 repositories vanished from BitBucket in 2020 when Mercurial support was phased out.[18,19]

While not strictly a requirement of an SBOM or component identification system, the ability to maintain long-lasting identities, references, and other artifacts is an important consideration. For example, SWHIDs use persistent, intrinsic (i.e., hash-based) identifiers for source code.

4.2.1.1 A Note on CPE

Common Platform Enumeration (CPE) is a widely-used software identification system. The NIST National Vulnerability Database (NVD) is currently one of the only public sources that maps supplier and component identity (using CPE) to vulnerability identity (using Common Vulnerabilities and Exposures, CVE[20]). CPE is widely used by many vulnerability management solutions and other tools that track publicly-known vulnerabilities. While the NVD currently uses CPE, the NVD also notes that "... SWID is being looked into as a possible replacement for CPE."[21]

4.3 Deconfliction

When searching for component identification data, SBOM authors may encounter multiple and conflicting answers. One actual component may have multiple identities, and the same identity may refer to different components. Changes in supplier organizations and component names are common causes of such identification conflicts. Suppliers go out of business or stop developing software, mergers and acquisitions happen, brand and component names change, and projects fork. Using a Java example, one supplier might use `com.sun.java` while another uses `com.oracle.java` based on when the suppliers started maintaining SBOM data. An identifier may not be available through a canonical source (such as a widely used package

[16] For more information about formats and tooling, see https://www.ntia.gov/SBOM.
[17] https://www.vice.com/en/article/ywen8x/13000-projects-ditched-github-for-gitlab-monday-morning.
[18] https://bitbucket.org/blog/sunsetting-mercurial-support-in-bitbucket.
[19] https://www.softwareheritage.org/2020/04/23/rescuing-250000-endangered-mercurial-repositories/.
[20] https://cve.mitre.org/.
[21] https://nvd.nist.gov/products.

manager), there may be multiple widely-used identifiers for a given component, or the same component may be identified in multiple repositories or package management systems.

In cases where it is not clear that two different identifiers represent the same or different components, SBOM authors or users should treat the components as different. When faced with multiple component identifiers, SBOM authors should select the identity that most closely aligns with their development and maintenance processes, such as choosing an identity based on the source of a component.

Having more component identification information (e.g., more of the baseline or extended attributes) can help disambiguate and resolve conflicts, but with the additional cost of managing and interpreting the information.

5 Supplier Identification

Identifying suppliers is a similar problem to identifying software components. There are multiple ways to identify suppliers. Multinational organizations may use different legal names in different countries, marketing brands may differ from legal or official organization names, and mergers, acquisitions, and other lifecycle changes impact supplier identification. The Japan Exchange Group,[22] D-U-N-S Numbers,[23] and IANA Private Enterprise Numbers[24] are just three of the many examples of existing namespaces that identify entities that may also be SBOM suppliers. As noted previously, various mechanisms based on DNS can be (and are) used for component identification. These mechanisms generally also support supplier identification, and a number of existing and developing SBOM formats and identification systems involve DNS. Both domain names and certificate authorities require registration with varying degrees of cost and validation.

In a highly-scalable SBOM model, dependency on any centralized source of data or administration raises concerns about resilience, cost (both to operate and to participating suppliers), and organizational bias. Despite these issues, a “supplier registry” model relying on a single global namespace for *suppliers* (not components) has been discussed. Conceptually, if supplier identities are globally unique, suppliers would have considerable discretion in how to identify components within their supplier namespaces. With or without a supplier registry, the SBOM model delegates component identification (and SBOM generation) to suppliers, who are likely the least cost avoiders.[25] The details of a supplier registry are beyond the scope of this paper, and further discussion is needed to assess the tradeoffs involved. Some key considerations include:

- SBOM model design: Supplier identity is globally unique, suppliers have significant autonomy in component identification;
- Cost to different sizes and types of suppliers to obtain and maintain registration;

[22] https://www.jpx.co.jp/english/.
[23] https://www.dnb.com/duns-number.html.
[24] https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers.
[25] https://www.lawfareblog.com/cybersecurity-and-least-cost-avoider.

- Supplier lifecycle, including merger, acquisition, and closure;
- Support for a variety of types of suppliers, including part-time, individual developers and open source software projects;
- Direct cost to operate the registry;
- Resilience and longevity of the registry, including distributed or decentralized mechanisms; and
- Organizational and political bias in the operation of the registry.

One notable concern is whether and how a supplier registry would practically work for the numerous open source software projects who are SBOM suppliers but who often lack a separate, non-person legal entity that can be registered. It would likely be impractical to require the creation of such legal entities or domain names for each supplier in real-world supply chains.

Pending further investigation, a supplier registry model might be a significant part of the solution to global component identification, particularly if existing supplier identification systems can be leveraged.

6 Conclusion

Even without a single, robust, and comprehensive global component identification solution, the guidance in this document can be applied now and is designed to drive convergence as SBOM becomes more widely adopted. Further work is needed to design, test, and implement global software component and supplier identification. A globally-scalable model will almost certainly involve concepts from distributed and federated systems (leveraging existing identification systems such as DNS) and require intrinsic component attributes (e.g., cryptographic hashes).