Framing Software Component Transparency: Establishing a Common Software Bill of Material (SBOM)

NTIA Multistakeholder Process on Software Component Transparency
Framing Working Group
2019-11-12

Table of Contents

1 Problem Statement

Modern software systems involve increasingly complex and dynamic supply chains. Lack of systemic visibility into the composition and functionality of these systems contributes substantially to cybersecurity risk. It also increases costs of development, procurement, and maintenance. In our increasingly interconnected world, risk and cost impact not only individuals and organizations, but also collective goods like public safety and national security.

Increased supply chain transparency can reduce cybersecurity risks and overall costs by:

- Improving the ability to identify vulnerable software components that contribute to cybersecurity incidents
- Reducing unplanned and unproductive work due to convoluted supply chains
- Allowing vendors that support transparency to more easily differentiate themselves in the market
- Reducing duplication of effort by standardizing formats across multiple sectors
- Facilitating the identification of suspicious or counterfeit software components

Achieving software supply chain transparency can increase trust and trustworthiness while lowering costs of our digital infrastructure. Individual pockets of people, policy, process, and technology are solving parts of the problem, but not in a systematic and scalable way that crosses development environments, product lines, vendors, sectors, and nations. A more systematic and collaborative approach can help.

To address the problem of poor software supply chain transparency, the National Telecommunications and Information Administration (NTIA) convened a multistakeholder process.[1] This document is an output from the Framing working group.

1.1 Goals

To achieve greater supply chain transparency, the primary goal of the Framing working group is to create a model for software component information that can be universally and transparently shared across industry sectors. The model defines and describes an SBOM: a software bill of materials. The model addresses relationships between components, the creation and sharing of SBOMs, the roles of participants, and SBOM integration with supply chains.

To scale this model globally, it is necessary to address the difficult problem of universally identifying and defining certain aspects of software component. So a subsidiary goal was to select a core, baseline set of attributes necessary to identify components with sufficient relative uniqueness. Another goal was to capture SBOM applications and consider what additional, optional attributes and external elements might be needed beyond the baseline set.

[1] https://www.ntia.doc.gov/SoftwareTransparency

Further background on this multistakeholder process and the Framing working group can be found in Section 6.

2 What is an SBOM?

An SBOM is effectively a nested inventory, a list of ingredients that make up software components. An SBOM identifies and lists software components, information about those components, and supply chain relationships between them. The amount and type of information included in a particular SBOM may vary depending on factors such as the industry or sector and the needs of SBOM consumers. For this initiative, the focus will be on establishing a minimum expectation for creating a baseline SBOM that outlines the minimum amount of information and process required to support basic and essential features.

Starting with a baseline set of information allows this process to be adopted by a variety of stakeholders quickly and then be further developed over time. This is one of the major drivers for establishing such a basic set of information as a starting point, rather than initially requiring a more robust set of attributes that may require more time and resources to collect and maintain. Beyond the core or minimum baseline SBOM, additional information may be needed as further development and practice matures within different sectors. An SBOM also needs to relate each component back to other components through the supply chains that compose software systems. Capturing and exchanging these links between components is an important feature of an SBOM.

Structured data formats and exchange protocols are another key characteristic of a functional SBOM, because they enable machine-readability and automation. Large SBOM consumer organizations will need to collate and manage large amounts of data from different suppliers, so it is critical to allow the consumers of this data to manage this in a machine-readable format for efficiency and expandability. Choosing a specific data format is an important part of this functionality. The Standards and Formats working group was established, in part, to address this issue. Without a specific data format and identification scheme, it would be nearly impossible to identify, track, and manage components that are named in an ad hoc fashion.

SBOMs do not exist as independent entities, completely isolated from other data sources. For example, the use of SBOM in vulnerability management requires a catalog of known vulnerabilities (e.g., CVE[2]), associations of vulnerabilities to components (e.g., NVD[3] use of CPE[4]), and possibly a means by which to convey the transitive exploitability or exposure of a vulnerability along supply chains. The use of SBOM for license management requires that licenses and their restrictions are mapped to components.

[2] https://cve.mitre.org/
[3] https://nvd.nist.gov
[4] https://nvd.nist.gov/products/cpe

A note about terminology: *Elements* are constituent parts of an overall *SBOM system*. *Components* are units of software and *attributes* are information about components. An *SBOM entry* identifies a component and its associated attributes. An SBOM is a collection of one or more SBOM entries. More terms are defined in Section 5.

2.1 SBOM Elements

The NTIA Software Component Transparency multistakeholder process reviewed existing software identification formats and thoroughly debated and questioned which elements would be necessary to build a functional SBOM system. Many of the answers depend on the desired use cases and applications that can be built on top of sufficient amounts and quality of baseline SBOM data. Without a way to systematically and consistently define and identify software components and their relationships, none of the desired applications are possible at scale. Therefore, one required element of an SBOM system is a baseline set of attributes to identify components.

2.2 Baseline Component Information

The primary purpose of an SBOM is to uniquely and unambiguously identify components and their relationships to one another. In order to do so, some combination of the following baseline information is required. It is possible that not every SBOM entry will require or be able to provide each of the baseline attributes. Certain attributes (e.g., Component Hash) provide greater uniqueness or unambiguity, as does having a greater number of the baseline attributes in an SBOM entry.

2.2.1 Author Name

Author of the SBOM entry (this may not always be the supplier).

2.2.2 Supplier Name

Name or identity of the **supplier** of the component in the SBOM entry, including some capability to note multiple names or aliases. When the author and supplier are the same, the supplier is defining a first-party authoritative component. When author and supplier are different, the author is making a claim or assertion about a component from a different supplier (see Section 2.4.1). The confidence in this assertion is not specified.

2.2.3 Component Name

One or more **component** name(s), including some capability to note multiple names or aliases. Component names can convey supplier names. Component (and supplier) names can be conveyed using a generic **namespace:name** construct.

2.2.4 Version String

Version information helps to identify a component.

It is not specified how authors, suppliers, or components are identified and named. Syntax is also not specified for version information, other than expecting basic consistency and logic (e.g., Semantic Versioning[5]).

2.2.5 Component Hash

Adding a cryptographic **hash** of the component is the most precise way to identify a binary, as-built component in an SBOM.[6] The hash is effectively the unique identifier of a component, however other baseline identification information will be useful and even necessary. Cryptographic signatures can be used in place of a hash, but add the complexities of key distribution and signature verification.

There exist techniques to create hashes of groups of components. For example, Software Package Data Exchange (SPDX) specifies a Package Verification Code that creates a hash of hashes for individual file components.[7]

2.2.6 Unique Identifier

A **unique identifier** can be generated and used to help identify components. This identifier could be a version 4 or 5 UUID.[8]

2.2.7 Relationship

Relationship is inherent in the design of the SBOM. The default relationship type is *includes*. To provide a more clear and consistent representation of relationships, this document inverts the direction of the relationship to be *included in*. The choice of direction is not important as long as one direction is chosen and used consistently.

[5] https://semver.org
[6] https://hal.archives-ouvertes.fr/hal-01865790/file/main.pdf
[7] https://spdx.org/spdx-specification-21-web-version#h.2p2csry
[8] https://en.wikipedia.org/wiki/Universally_unique_identifier

Using the example from Section 2.5, the following statements are equivalent:

1. Acme Application v1.1 *includes* Bob's Browser 2.1.
2. Bob's Browser v2.1 is *included in* Acme Application v1.1.

It is possible to further refine the *included in* relationship, as supported by both SPDX[9] and the software identification (SWID) tag format.[10] The relationship also allows an SBOM author to express their understanding of additional relationships for components that lack SBOM information and for which the author is not the supplier. See Section 2.4 for additional context.

2.3 Mapping to Existing Formats

Table 1 (recreated from Table 1 in *Survey of Existing SBOM Formats*[11]) maps baseline component attributes across SPDX and SWID. More formats and their mappings are described in later in the *Survey* document.

Baseline	**SPDX**	**SWID**
Supplier Name	`(3.5) PackageSupplier:`	`<Entity> @role (softwareCreator/publisher), @name`
Component Name	`(3.1) PackageName:`	`<softwareIdentity> @name`
Unique Identifier	`(3.2) SPDXID:`	`<softwareIdentity> @tagID`
Version String	`(3.3) PackageVersion:`	`<softwareIdentity> @version`
Component Hash	`(3.10) PackageChecksum:`	`<Payload>/../<File> @[hash-algorithm]:hash`
Relationship	`(7.1) Relationship: CONTAINS`	`<Link> @rel, @href`
Author Name	`(2.8) Creator:`	`<Entity> @role (tagCreator), @name`

Table 1: Mapping baseline component information to existing formats

[9] https://spdx.github.io/spdx-spec/7-relationships-between-SPDX-elements/
[10] https://csrc.nist.gov/publications/detail/nistir/8060/final
[11] https://www.ntia.gov/SBOM

2.4 Component Relationships

The first step in developing an SBOM is often to start with a simple list of first-level components. However, in order to scale effectively, an SBOM needs to capture nested supply chain relationships between components to the extent that those relationships are known. Bills of materials for physical components often describe these relationships as a "Multi-level BOM."[12] While an SBOM may account for many types of relationships, the baseline described in Section 2.2.7 defines a single type of relationship: *includes* (or *included in*). In other words, an upstream or child component is *included in* a downstream or parent component. In Figure 1, Bingo Buffer is upstream of Acme Application, Bingo is an upstream supplier to Acme. The words "included" and "upstream" are synonymous. Other types of relationships, in addition to the required *included in,* may be necessary or useful. Existing SBOM formats support different types of relationships today.

While subcomponents are intended to provide some dependent functionality of the superior or parent component, it is common for parts of the subcomponent to not be used. For example, a software program might include a library subcomponent, but only use some of the functions provided by the library. This becomes important in some SBOM applications, particularly vulnerability management. If a vulnerability affects a subcomponent, the inclusion of that subcomponent in a parent component may or may not transitively cause the parent component to be vulnerable.[13]

An SBOM is made up of one or more components, and each component may effectively be another SBOM, also containing one or more components. An SBOM must include at least one primary component, and this component defines and identifies both the component and the SBOM. Every other component listed in an SBOM has the *included in* relationship.

Component relationships are illustrated in Figure 1 and Table 2.

2.4.1 Knowledge About Relationships

Ideally, every supplier will create and provide SBOMs for their components, and all consumers will obtain complete chains of these authoritative SBOMs. For every component, Author Name (2.2.1) will equal Supplier Name (2.2.2), and in this ideal world, there will be perfect knowledge. Until this state of transcendent SBOM utopia is achieved, SBOM authors may want to make non-authoritative claims or assertions about SBOMs for which the authors are not the suppliers. One expected case is that a supplier wants to assert their belief about upstream components for which an authoritative SBOM does not exist.

[12] https://medium.com/@openbom/openbom-fundamentals-all-about-openbom-multi-level-boms-f06f50ca7f74
[13] https://www.ntia.doc.gov/files/ntia/publications/wysopal_swct_kickoff_perspective.pdf

These relationship assertions can be recorded by an author using an additional component attribute. The following four categories cover the range of an author's knowledge about another supplier's components.

1. **Unknown**. This is the default. There is not yet any claim, knowledge, or assertion about upstream components. Immediate upstream components are not currently known and therefore not yet listed, or there may not be any upstream components. This default value implies the open-world ontological assumption.[14]
2. **Root**. There are no immediate upstream relationships. As defined by the supplier, the component has no subcomponents.
3. **Partial**. There is at least one immediate upstream relationship and may or may not be others. Known relationships are listed.
4. **Known.** The complete set of immediate upstream relationships are known and listed.

Relationship assertions apply to a component and describe its immediate upstream relationships. Figure 2 and Table 4 add relationship assertions to the examples in Figure 1 and Table 2.

Assertions about upstream relationships support at least two types of analysis or interpretation. In the first, more limited, interpretation, knowledge about Acme Application v1.1 is treated as Known, because all immediate upstream subcomponents are known. This allows a supplier or author to convey that they have provided a comprehensive list of immediately upstream components. In the second interpretation, Acme Application v1.1 is treated as Partial because some of its upstream components are Partial or Unknown. Both interpretations can be useful. However, it is important to recognize the limited scope of the first.

2.5 SBOM Examples

To further illustrate the relationships described in the previous section, consider these SBOM examples. Figure 1 and Table 2 show two different ways in which to view SBOM information and relationships. These are conceptual representations and not specific formats like SPDX or SWID.

[14] https://en.wikipedia.org/wiki/Open-world_assumption

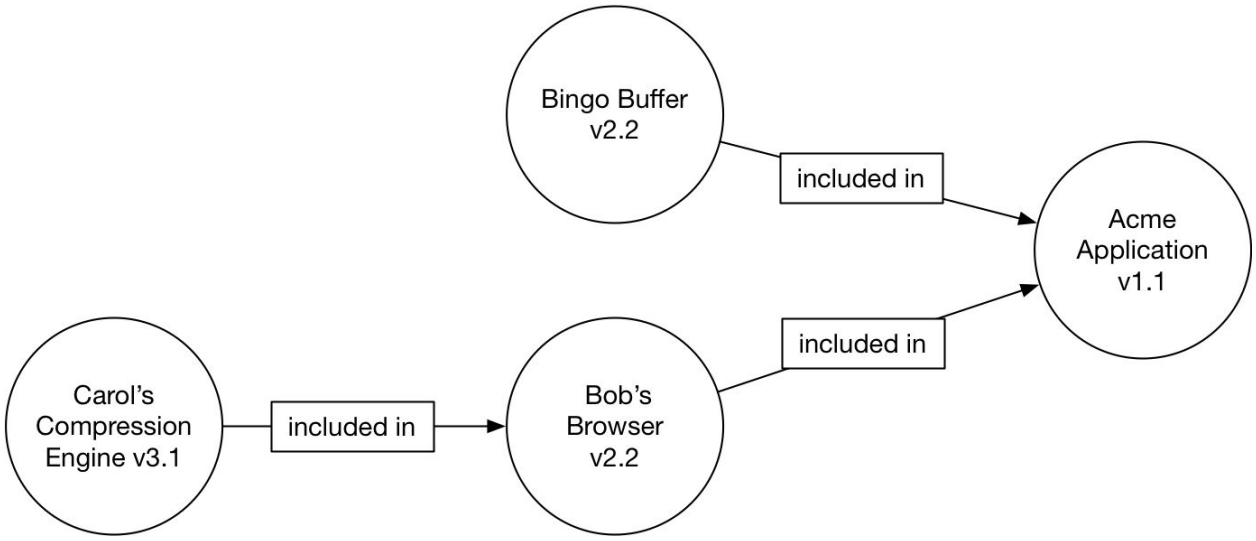

Figure 1: Conceptual SBOM tree

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship
Application	Acme	1.1	Acme	0x123	234	Self
|--- Browser	Bob	2.1	Bob	0x223	334	Included in
|--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in
|--- Buffer	Bingo	2.2	Acme	0x423	534	Included in

Table 2: Conceptual SBOM table

In the simplest case of a root SBOM or component, consider a single component that was created entirely from scratch with no dependencies: Carol's Compression Engine v3.1. The SBOM of this component consists of only one SBOM entry. This single entry defines the component and the SBOM and has a special relationship type of "self." This example is shown in Table 3.

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Assertion
Compression Engine	Carol	3.1	Carol	0x323	434	Self	Root

Table 3: Conceptual SBOM table for root component

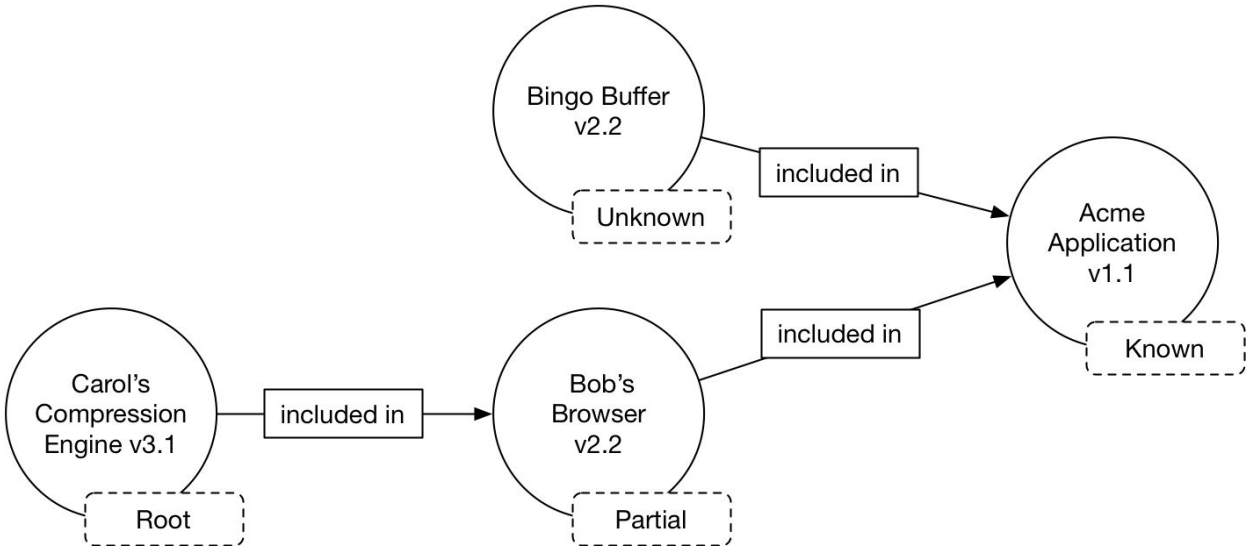

Figure 2: Conceptual SBOM tree with upstream relationship assertions

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Assertion
Application	Acme	1.1	Acme	0x123	234	Self	Known
|--- Browser	Bob	2.1	Bob	0x223	334	Included in	Partial
|--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in	Root
|--- Buffer	Bingo	2.2	Acme	0x423	534	Included in	Unknown

Table 4: Conceptual SBOM table with upstream relationship assertions

In Figure 1 and Table 2, the SBOM for Acme Application has four components, since each SBOM includes itself. Acme makes a component named “Application” that uses exactly two third party components, Bob’s Browser and Bingo Buffer. Bob’s browser, in turn, uses Carol’s Compression Engine, and may also use other components. Carol’s Compression Engine does not include further subcomponents, while Bingo Buffer may or may not have any further dependencies.

2.6 Additional Elements

In addition to baseline information, an SBOM likely requires additional elements and component attributes to support different use cases or applications. The specific information needed depends on the use case or application, and not all additional information will support each use

case or application. Section 4.6 describes SBOM applications and some of the additional information needed for those applications.

2.6.1 Authenticity

An SBOM system must support the ability to cryptographically authenticate SBOM information. In general, this means that an author must be able to digitally sign an SBOM. Authentication is optional; that is, an author is not required to sign an SBOM. Authentication requires appropriate digital signature and public key infrastructure.

3 Data Formats

Much of the benefit of software component transparency is easier to realize if SBOM data can be generated, shared, and processed by machine. In practice, machine processing and automation will be necessary. This requires widespread interoperability across the supply chain which, in turn, requires standardized data formats and identification schemes.

Stakeholders formed a working group to enumerate and review existing standards and formats. No single widely used standard was explicitly designed to only address SBOM use cases. However, the working group identified two formats in widespread use: SPDX, an open source machine-readable format stewarded by the Linux Foundation; and SWID, an international XML-based standard used by commercial software publishers and adapted by NIST. Both of these standards cover baseline SBOM information (see Section 2.3).

Both formats focus on the core problem of identifying software entities and conveying associated metadata, and have the requisite fields to cover the needs for the baseline SBOM. While different communities can select the format that meets their needs, the baseline SBOM data can be represented across the supply chain through straightforward translation. For more information about SPDX, SWID, and other existing inventory and SBOM formats, please see the *Survey of Existing SBOM Formats and Standards*.[15]

[15] https://www.ntia.gov/SBOM

4 SBOM Processes

This section describes how to create and exchange SBOM information from from three stakeholder perspectives: those who produce, choose, and operate software. These perspectives are described in detail in *Roles and Benefits for SBOM Across the Supply Chain*.[16] Two SBOM applications—license management and vulnerability management—are explored in further detail to illustrate SBOM as an independent data source as well as integration into typical business processes.

4.1 SBOM Creation: How

To create an SBOM, the supplier defines components that the supplier creates themselves, produces baseline component information and any additional attributes for those components, and enumerates the list of all directly included components. SBOM information will ideally be generated as an integral part of the supplier's software build and packaging processes, which can be accomplished with modifications to existing development tools.

Any entity creating, modifying, packaging, and delivering software or software systems is considered to be a supplier and is therefore responsible for defining components and creating SBOMs. This includes system integrators who are essentially considered suppliers for SBOM purposes. An organization can also act as a supplier for internally developed components.

When SBOMs for included components are available from upstream suppliers, those SBOMs are provided with or added to the primary SBOM. Where such information is not available, a supplier can provide "best effort" SBOMs, which will be indicated by the fact that the author for an included component SBOM will not be the same as the supplier of the component. Section 2.4.1 describes a way for SBOM authors to make assertions about indirectly included upstream components for which the supplier has not provided an SBOM.

An SBOM includes attributes used to identify components and additional attributes to capture characteristics of or information about components. Identity attributes are required, and additional attributes may or may not be required depending on the use case or application.

An SBOM from the component's supplier serves as a system of record or authoritative source of information about the component. As noted elsewhere, some information may need to be validated with other external sources. For example, vulnerability information about a component can sometimes be derived from the National Vulnerability Database (NVD) using the Common Platform Enumeration (CPE).

[16] https://www.ntia.gov/SBOM

4.2 SBOM Creation: When

A new SBOM should be created for every new release of a component. Changes to components require corresponding changes to SBOMs. Changes to components are often noted as updates, upgrades, releases, and patches. Ideally, changes to components are indicated by a change in the baseline version string attribute. An SBOM should be created or updated when information about included components changes, even if the components themselves have not changed. For example, an SBOM should be updated when an included component is changed and also when a supplier starts providing or provides new SBOM information. In the former case, the underlying software component has changed; in the latter, the component has not changed, but new SBOM information is available. Maintaining current SBOM information is essential.

When changing (including patching or updating) an existing component, it is possible to treat the change itself as a separate, new component added to the existing SBOM or to create a new component, ideally with a new version string. In the example from Table 5, Bob's Browser v1.1 with update 37 is equivalent to Bob's Browser v1.1.1. SBOM authors should use one method consistently.

Timeline	Additional component (A)	New version string (B)
Before change	Bob's Browser v1.1	Bob's Browser v1.1
After change	Bob's Browser v1.1 Bob's Browser update 37	Bob's Browser v1.1.1

Table 5: Patch or update options

4.3 SBOM Exchange

It is necessary to exchange SBOM information. The primary exchange is directly from a supplier to a consumer through a single downstream supply chain link. As part of delivering the component, the supplier also delivers the SBOM, or a means by which the user can easily obtain the SBOM, such as a URL or other reference. This direct delivery does not preclude aggregation or cataloging of SBOM information by suppliers, consumers, or others.

Due to the variety of different software and device ecosystems, it is unlikely that one SBOM exchange mechanism will suffice. Some existing formats, namely SWID and SPDX, are provided as additional files as part of a component distribution or delivery. For devices with storage and power constraints, an option is to provide a URL with a unique ID to look up SBOM information on a supplier's website. Dynamic access to an SBOM may be a good option for

such devices as well. Protocols such as ROLIE,[17] OpenChain,[18] and ATOM[19] can be leveraged to exchange large amounts of information with the ability to collect it on-demand and in an efficient manner. The availability of SBOM using some of these dynamic protocols (for environments such as software development environments to asset management systems) will be a key to continued adoption of the SBOM.

4.4 Network Rules

Participants in an SBOM system include suppliers creating SBOM information and consumers receiving it. In many cases, one entity will be serving both these functions, as software is often made of other software. Participants follow a set of network rules to ensure SBOMs are both viable and useful to all parties in the ecosystem.

An SBOM lists components that can be:

1. Originally created *de novo* by a supplier who is the authoritative source of the software
2. Integrated as a component from an upstream supplier who also provides an SBOM
3. Integrated as a component from an upstream supplier who does not provide an SBOM

Suppliers create SBOMs for the components the suppliers define. As described in the earlier sections on component relationships, an SBOM must list at least one primary component, identifying the SBOM itself as a component.

As part of delivering components to users, the supplier also delivers the associated SBOM(s), or provides a means for the consumer to easily obtain SBOMs. SBOMs include both components that the supplier originally creates and components that the supplier obtains from other suppliers.

In such a structure (likely a directed acyclic graph, as shown in figures 1, 2, and 3), ultimate upstream or root suppliers only create original components and do not include components (that is, do not have dependencies) from any other supplier. In Section 2.5, Carol is an example of a root supplier. Components flow downstream from these root suppliers throughout the graph. At the far right of the tree, leaf operators obtain only components and SBOMs; they do not produce components or create SBOMs. Throughout the middle of the tree, most participants act as both suppliers and consumers. Even end-user organizations may act as suppliers, producing SBOMs for in-house components or external components such as websites, mobile applications, or devices.

Suppliers are responsible for components they create and those they include. Suppliers are also responsible for providing the collected set of components to their downstream consumers. In a macroeconomic sense, suppliers are the least cost avoiders, since they have high quality

[17] https://tools.ietf.org/html/rfc8322
[18] https://www.openchainproject.org
[19] https://tools.ietf.org/html/rfc4287

authoritative information about their components and low costs to generate and share that information.[20] This model also distributes the cost to produce SBOM information across software supplier markets.

In this network, there are some scenarios where a supplier may create SBOMs for third-party components where the supplier is acting as the author of the SBOM and not the creator of the component. A supplier can create as many third-party SBOMs as they choose, with or without dependency relationships. When a supplier creates such SBOMs, the supplier is expected to be be clear that they are the author only of the SBOM and are not the supplier of the component. This informs consumers of the lack of first-hand authoritative SBOM information. In such a case, the Author Name (2.2.1) and Supplier Name (2.2.2) would be different. This knowledge about upstream relationships described in further detail in Section 2.4.1.

The concepts around SBOM exchange and network rules are designed so that those who choose and operate software can obtain comprehensive lists of components they use across different suppliers and supply chains. Figure 3 expands the example in Figure 2 to show an operator who uses two software components from two different supply chains. The operator has two SBOMs, one in Table 4 and one in Table 6.

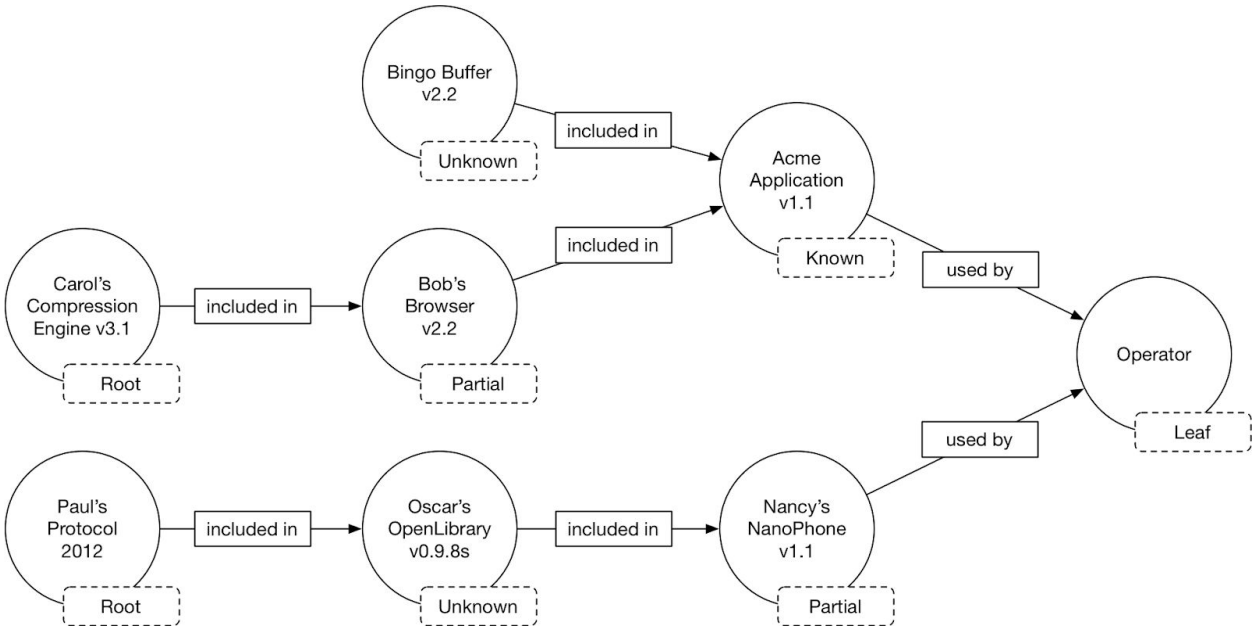

Figure 3: Operator tree with two supply chains

[20] https://www.lawfareblog.com/cybersecurity-and-least-cost-avoider

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Assertion
NanoPhone	Nancy	1.1	Nancy	0x523	237	Self	Partial
|--- OpenLibrary	Oscar	0.9.8s	Nancy	0xA23	394	Included in	Partial
|--- Protocol	Paul	2012	Nancy	0xB53	934	Included in	Root

Table 6: Conceptual SBOM table representation for Nancy's NanoPhone

4.5 Roles and Perspectives

4.5.1 Perspectives

Different stakeholders will use SBOMs in complementary yet distinct ways. *Roles and Benefits for SBOM Across the Supply Chain* presents three stakeholder perspectives: those who produce, choose, and operate software.

4.5.1.1 Produce

Producers of software are typically, but not always, suppliers. A goal of this SBOM system is that all producers create SBOMs for their components; however, there may be cases in which one supplier authoring an SBOM will have to create an SBOM for a different supplier's component.

While some suppliers may be reluctant to share SBOM information with their customers, there is no doubt that suppliers benefit from receiving SBOMs associated with upstream components that are included in their products. In the case where a supplier does not provide SBOM information about subcomponents, there is a higher likelihood that this lack of clarity will cause downstream users to assume the worst about the unknown parts of the product. An additional benefit to suppliers is the ability to determine which organization to contact to get fixes for vulnerabilities in upstream components.

A key problem occurs when vulnerability scanners run by a supplier or the supplier's customers indicate that fixes are missing for vulnerable components. Unless the components are all directly included, the supplier does not know which directly included components include the subcomponents that are missing vulnerability fixes. SBOMs solve this problem when the full SBOM hierarchy is available since that will indicate which directly included component suppliers need to be contacted to provide vulnerability fixes.

Part of a vulnerability management application of SBOM could include the ability to convey how a vulnerability in an upstream component affects the downstream component. If the SBOM indicates that the vulnerability in the upstream component cannot be exploited through the downstream component, then the supplier of the downstream component knows that no fixes

are necessary. This is a huge benefit for both suppliers, who will not have to create fixes, and their customers, who will not have to apply them.

4.5.1.2 Chose

SBOMs can be used by prospective choosers (e.g., acquisition or procurement functions) considering the use of a component or product that has an associated SBOM. These users are likely to be interested in information directly attributable to the product, such as its baseline component or license information. Associated SBOM information about licensing, vulnerabilities, and support lifecycle can factor in to the selection process.

In the license management use case, the prospective user has visibility into the license of every constituent component for a Product under consideration for use in addition to the Product itself. This will give the user greater awareness when choosing a product and its appropriate license arrangement for the business need or application.

In the vulnerability management use case, the prospective user may use the baseline component information to identify the history of published vulnerabilities for a product and the timeliness of the vendor's response to these vulnerabilities. This allows the chooser to make an informed decision with consideration given to the product's security lifecycle.

4.5.1.3 Operate

In any industry, "Operations" suffers primarily due to a lack of complete information on components or products they are expected to support. An SBOM becomes a very relevant source of this information to provide visibility into the software and its components. Some of this information may be static, such as licensing information. However, due to the dynamic nature of software, some of this information may change or be updated after a product's initial distribution.

Most of the information of ongoing interest for "Operators" is expected to be found in SBOM updates. Operators can also use the current information to verify the state of the software before it is to be in production at their site or business.

4.6 Applications of SBOMs

The core focus of the SBOM is on identifying components and their relationships. Many of the other applications of its use require additional information beyond what is captured in the baseline SBOM, such as new attributes or external data connections, in order to function. In this section several notable applications of the SBOM have been highlighted:

1. **Vulnerability Management:** Vulnerability management is one of the more prominent applications. Today, it is often an expensive and time-consuming effort to determine whether a vulnerable subcomponent is used, and if the vulnerability transitively makes the downstream component vulnerable or exploitable. SBOM data helps suppliers, users, and other defenders more quickly and accurately assess the risk posed by

vulnerable components otherwise hidden behind supply chain relationships. It works with the principle that you cannot protect what you do not understand.
Additional information needed: sources of vulnerability information such as CVE and the NVD.

2. **Intellectual Property:** There are a number of intellectual property applications that could be improved with better inventory data. Managing software licensing (including constraints on use or redistribution) for included components, and tracking entitlement (permission to use copies or features of components) are some of the existing applications. A notable market exists for software composition analysis tools to help determine the contents of components. SBOM data would improve intellectual property applications.
   Additional information needed: associations of different licenses and types of licenses to components, and a way to evaluate the net effect of different components with different licenses combined into an assembled good. In fact, both SPDX and SWID were designed to carry license information.
3. **High Assurance:** High assurance of the source and integrity of components. Requires further information about suppliers, how components are built, the chain of custody as components move through the supply chain, and how any modifications are made.
   Additional information needed: information about the **pedigree** and **provenance of components**, such as how they were built.

In addition to the specific additional applications identified above, other sets of information could be useful to the stakeholders creating and using SBOMs, including:

- **End-of-Life Dates** and the ability to indicate what **technologies** a component supports would be useful for multiple applications.
- **Implemented Technologies.** Groups of components (around technologies or other concepts) could be implemented through the proposed recursive relationship model, treating technology as an upstream component. For example, “component X implements DNS” allows a user to identify all DNS-relevant components.

4.7 Tool Support

The availability of SBOM generation and management tools will be critical for widespread adoption. Tools are available for some existing formats, e.g., swid-tools,[21] swidGenerator,[22] and SPDX Tools.[23] SBOM functionality will need to be integrated into software development and asset management systems.

[21] https://apps.fedoraproject.org/packages/swid-tools
[22] https://github.com/strongswan/swidGenerator
[23] https://github.com/spdx/tools

5 Terminology

The following terms have specific meaning within the scope of this document and within the overall multistakeholder process. Each definition is written to be a direct gramatical replacement for the term.

5.1 SBOM

(Software Bill of Materials)

list of one or more identified components and other associated information

The SBOM for a single component with no dependencies is just the list of that one component. “Software” can be interpreted as “software system,” thus hardware (true hardware, not firmware) and very low-level software (like CPU microcode) can be included. The primary focus of this effort is software components; however, hardware is not excluded.

5.2 SBOM System

set of elements and processes acting together to provide the ability to create, exchange, and use SBOMs

5.3 Element

part of an SBOM system

5.4 Component

unit of software defined by a supplier at the time the component is built, packaged, or delivered

A product is a component. So is a library. So is a single file. So is a collection of other components, like an operating system, office suite, database system, car, an engine control unit (ECU) in a car, a medical imaging device, or an installation package. In SPDX terms “package,” “file,” and “snippet” map to “component.” In SWID terms: “<softwareIdentity> @name.” While “component” is defined and intended to represent compiled or binary code, source code is not excluded, nor is hardware.

5.5 Attribute

characteristic of or information about a component

Baseline attributes are defined in Section 2.2, other attributes can be defined as needed to meet specific use cases and applications. In a table, an attribute is a column.

5.6 SBOM Entry

component and its associated attributes

In a table, an entry is a row.

5.7 Author

entity that creates an SBOM

Note: When author and supplier are different, this indicates that one entity (the author) is making claims about components created or included by a different entity (the supplier).

5.8 Supplier

entity that creates, defines, and identifies components and produces associated SBOMs

A supplier may also be known as a manufacturer, vendor, developer, integrator, maintainer, or provider. Ideally, all suppliers are also authors of SBOMs for the suppliers' components. Most suppliers are also consumers. A supplier with no included upstream components is a root entity.

5.9 Consumer

entity that obtains SBOMs

An entity can be both a supplier and consumer. An "end-user" consumer (that is not also a supplier) may also be called an operator or a leaf entity.

6 Background

6.1 Overview of the NTIA Multistakeholder Process

On July 19, 2018, the National Telecommunications and Information Administration (NTIA) convened a meeting of stakeholders from across multiple sectors to begin a discussion about software transparency and the proposal being considered for a common structure for describing the software components in a product containing software. The output of this meeting was to create a number of task groups, which then led to documents produced by each of these groups. This document is the output of the "Understanding the Problem" (or "Framing") task group and seeks to (1) describe the problems that are initiating the need for a software bill of materials, (2) identify what a baseline SBOM should include, and (3) provide an overview of the processes to manage SBOMs. The reports from the other task groups are available from the NTIA website.[24]

Understanding the Problem (Framing)
Describe the scope of the idea of software transparency and the problems it seeks to solve, including how SBOM data might be shared. Outputs included (1) a description of a baseline SBOM elements, (2) the identification of goals, problem statement, and scope, (3) useful terminology, and (4) an outline of basic process and implementation guides.

Use Cases and State of Practice
Focused on identifying use cases, current and possible future, where SBOMs or similar data is used to achieve various goals. Through review of the current state of practice, outputs were developed to identify what works today and describe barriers to success.

Standards and Formats
Investigated existing standards and initiatives as they apply to identifying the external components and shared libraries, commercial or open source, used in the construction of software products. The group analyzed efforts underway in the community and industry related to assuring this transparency is readily available in a machine-readable manner.

Healthcare Proof of Concept
A collaborative effort between healthcare delivery organizations and medical device manufacturers that established a prototype SBOM format and exercised use cases for SBOM production and consumption. The goal was to demonstrate successful use of SBOMs and relate to the overall cross-sector effort to establish standardized formats and processes.

[24] https://www.ntia.gov/SBOM

6.2 Mission Statement

The mission of the NTIA multistakeholder process on Software Component Transparency is to:

> Explore how manufacturers and vendors can communicate useful and actionable information about the third-party and embedded software components that comprise modern software and IoT devices, and how this data can be used by enterprises to foster better security decisions and practices.[25]

The goal of this process is to foster a market offering greater transparency to organizations, who can then integrate this data into their risk management approaches.

6.3 Scope

The scope of this initiative will include the development of a model for a Software Bill of Materials (SBOM), how it can be shared, and how it can be used to help foster better security decisions and practices. To make the SBOM useful, this initiative will also need to outline the applicable use cases to ensure that the output is useful for all stakeholders.

All industries utilizing or producing software should be considered in the scope of this initiative, including automotive, financial, healthcare, operational technology (OT), and “traditional” IT. The focus is on software, the “S” in SBOM. Despite the focus on software and not hardware, software doesn’t run by itself. A software system requires not only traditional computing hardware (e.g., CPUs, memory, disk, network, etc.,) but may also include functional hardware that makes devices actually work, such as actuators and sensors.

This effort will focus on a limited scope of addressing the creation of harmonized elements of an SBOM that will aid in the sharing of software component information. There are, however, related dependencies and supporting activities that should be considered and will need to be addressed outside of this scope in order to fully realize a holistic solution to the need for software transparency:

- Lists of known vulnerabilities (e.g., CVE)
- Mapping vulnerabilities to components
- Conveying the transitive exploitability or exposure of vulnerabilities through supply chains
- Standardized sharing mechanism for SBOMs

Learning from principles of supply chain management in other fields, this effort is focused on harmonizing how to describe software components in the form of an SBOM. The intention is to improve how information is shared about the software that we build, acquire, operate, and depend on.

[25] https://www.ntia.doc.gov/SoftwareTransparency

7 Conclusion

Organizations across the globe face operational and supply chain questions about the software being actively used in their environments. Much of this software handles critical portions of their business activities while providing very little or no visibility into the software's component parts. Questions around known vulnerabilities continually go unanswered due to this lack of visibility. One way to increase the transparency and enable businesses to better manage the security of their networks and allow software vendors to monitor their products is to establish a harmonized model for creating and exchanging SBOMs.

To be useful to end-user organizations, an SBOM needs to include baseline identity and relationship information that enables correlating and connecting software components as they move through the supply chain. In the interest of rapid adoption, a set of minimum baseline attributes have been defined. These attributes generally align with existing formats such as SPDX and SWID. As noted in this document, however, limiting an SBOM to only this baseline information is not sufficient to enable a number of identified use cases and applications.

While the use of SBOMs will not solve all of the security issues facing the industries involved in this initiative, it will help to increase transparency and better inform those defending their networks and systems from known vulnerabilities. As this space matures, the ready availability of baseline SBOM information will hopefully lead to further work in establishing more coordinated and standardized methods of sharing and managing SBOMs. One of the reasons to standardize the structure and content of the SBOM is to enable these next steps. Tooling will also be a major factor in the adoption and further development of SBOMs.

Overall, the goal is to ensure that the necessary information, captured and exchanged through SBOMs, is available to those who need it, thereby leading to better asset, intellectual property, and vulnerability management.

8 Acknowledgements

The chairs of the Framing Working group, Michelle Jump (Nova Leah) and Art Manion (CERT Coordination Center), thank the working group membership for their time and effort. Acknowledgement does not imply endorsement of this document and its content.

Josh Corman
David Dillard, Veritas Technologies
Christopher Gates, Velentium
Les Gray, Abbott
Charlie Hart, Hitachi
Audra Hatch
Ed Heierman, Abbott
JC Hertz
Kent Landfield
Eliot Lear
Bruce Lowenthal, Oracle
Chujiao Ma
Bob Martin, MITRE Corporation
Chandan Nandakumaraiah, Juniper Networks
Vijay Sarvepalli, CERT Coordination Center
Duncan Sparrell, sFractal Consulting
Kate Stewart, The Linux Foundation

The working group also recognizes and appreciates the efforts of Allan Friedman and Megan Doscher of the National Telecommunications and Information Administration.