Framing Software Component Transparency

NTIA Multistakeholder Process on Software Component Transparency
Framing Working Group
2019-09-03

Éamonn Ó Muirí
https://flic.kr/p/46dsiz
https://creativecommons.org/licenses/by/2.0/legalcode

Table of Contents

Table of Contents 2

1 Problem Statement 4

1.1 Goals 4

2 What is an SBOM? 5

2.1 SBOM Elements 6

2.2 Baseline Component Information 6

2.2.1 Author Name 6

2.2.2 Supplier Name 6

2.2.3 Component Name 7

2.2.4 Version String 7

2.2.5 Component Hash 7

2.2.6 Unique Identifier 7

2.2.7 Relationship 7

2.3 Mapping to Existing Formats 8

2.4 Component Relationships 8

2.4.1 Knowledge About Relationships 9

2.5 SBOM Examples 9

2.6 Additional Elements 11

3 Data Formats 12

4 SBOM Processes 13

4.1 SBOM Creation: How 13

4.2 SBOM Creation: When 14

4.4 SBOM Exchange 14

4.5 Network Rules 15

4.6 Roles and Perspectives 17

4.6.1 Perspectives 17

4.6.1.1 Produce 17

4.6.1.2 Chose 18

4.6.1.3 Operate 18

4.7 Applications 18

4.8 Tool Support 19

5 Terminology 19

5.1 SBOM 19

5.2 Component 20

5.3 Author 20

5.4 Supplier 20

5.5 Consumer 20

6 Background 21

6.1 Overview 21

6.2 Mission Statement 22

6.3 Scope 22

7 Conclusion 23

8 Acknowledgements 24

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 as well as the costs of development, procurement, and maintenance. In our increasingly interconnected world, risk and cost impact not only individuals and organizations directly but also collective goods like public safety and national security.

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

- Enhancing the identification of vulnerable software components that contribute to cybersecurity incidents
- Reducing unplanned and unproductive work due to convoluted supply chains
- Supporting more informed market differentiation and component selection
- Reducing duplication of effort by standardizing formats across multiple sectors
- Identifying suspicious or counterfeit software components

thus increasing trust and trustworthiness while lowering costs of our digital infrastructure.

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.

1.1 Goals

To achieve greater supply chain transparency, the primary goal of the Framing working group was to create a standardized model for software component information that can be universally shared across industry sectors. The model defines and describes an SBOM, 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 every 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 use cases and applications and consider what additional, optional attributes and external elements might be needed beyond the baseline set. During the course of the multistakeholder process a handful of SBOM applications emerged. These included:

- Vulnerability management: Improving the ability to assess and mitigate risk caused by vulnerable components
- Intellectual property management: Verifying proper and legal use of components with different licensing and entitlement terms
- High assurance: Assessing confidence that components are authentic and unaltered
- Improved supply chain hygiene: Reducing complexity and costs, increasing quality

- End-of-life: Recognizing when components will no longer be supported

Keeping these applications in mind, the Framing working group chose not to dig deeply into them, but instead focus on the core model, baseline component identity, and other elements necessary and common to all of the applications.

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

2 What is an SBOM?

SBOM stands for "software bill of materials." An SBOM identifies and lists software components, information about those components, and supply chain relationships between them. The amount and type of related information included in a particular SBOM varies depending on specifics such as the industry using the SBOM and the demands of certain customers of the SBOM such as a government entity or large vendor. However, for this initiative the focus will be on establishing a minimum expectation for creating a baseline SBOM which, as the name suggests, outlines the minimum amount of information required in an SBOM for the most basic and essential usage of an SBOM to be practical.

An SBOM system should have a way of relating each component back to the broader instantiation. This may vary, depending on the vendor. If there are relationships of links amongst the components, this is another beneficial data point to add to the SBOM. However, some of this information may be added as the concept of an SBOM matures within the various industries and vendor processes. The value of quickly gathering this minimum set of baseline information for a majority of software components will significantly improve the ability for each industry to better manage the components that they use. Starting with a baseline set of information allows this process to be adopted by a variety of stakeholders quickly and then be built upon over time. This is one of the major drivers for establishing such a basic set of information as a starting point, rather than requiring a more robust set of attributes that may require more time and resources to collect and maintain.

Machine-readability is another key part of a useful SBOM system. Large organizations will need to collate and manage large amounts of data from various suppliers/vendors so it is critical to allow the users 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 and will be outlined later in the document (insert specific reference, as well as additional documents for more resources from NTIA project). Without a specific data and naming format, it would be near impossible to trend and manage components that are named in an ad-hoc fashion.

SBOMs will also not exist as independent entities, completely isolated from other data sources. For example, vulnerability management requires an external catalog of known vulnerabilities (e.g., CVE[1]), association of vulnerabilities to components, and possibly a means by which to convey the transitive exploitability or exposure of a vulnerability along supply chains. The user would map SBOM components to the catalog for further risk evaluation and mitigation.

2.1 SBOM Elements

The Framing working group and the overall multistakeholder process reviewed existing software identification formats and thoroughly debated and questioned which elements[2] would be necessary to build a functional, core 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 is a baseline set of attributes to identify components.

2.2 Baseline Component Information

The primary purpose of an SBOM record is to uniquely and unambiguously identify components and their relationships to each other. In order to do so, some combination of the following baseline information is required. It is possible that not every SBOM record will require or be able to provide each of the baseline attributes. Certain attributes (e.g., Component Hash) provide greater uniqueness or unambiguity, and so does having a greater number of the baseline attributes in an SBOM record.[3]

In addition to providing baseline component information, an SBOM must also support the ability to cryptographically authenticate SBOM information. SBOM authentication is optional, and requires appropriate digital signature and public key infrastructure.

2.2.1 Author Name

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

2.2.2 Supplier Name

Name or identity of the **supplier** of the component in the SBOM record, 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 third-party claim about a component from a different supplier. The confidence in this claim is not specified.

[1] https://cve.mitre.org/

[2] “Elements” are constituent parts of an overall SBOM system, as opposed to “components” (units of software) and “attributes” (information about components).

[3] An SBOM “record” is a component and its associated attributes.

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.

We do not specify how authors, suppliers, or components are identified and named. We also do not specify syntax for version information, other than pleading for some basic consistency and logic [semver.org].

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.[4] 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 are techniques to create hashes of groups of components, for example, SPDX specifies a Package Verification Code that creates a hash of hashes for individual file components.[5]

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.[6]

2.2.7 Relationship

Relationship is inherent in the design of the SBOM. The default relationship type is, broadly, “includes.” For example, in Table 2, Acme Software v1.1 includes Bob’s Browser v2.1. It is possible to further refine the “includes” relationship, as supported by both SPDX[7] and SWID.

The default “includes” relationship supports the following assertions about what the SBOM author believes about additional relationships. See Section 2.4 for additional context.

[4] https://hal.archives-ouvertes.fr/hal-01865790/file/main.pdf
[5] https://spdx.org/spdx-specification-21-web-version#h.2p2csry
[6] https://en.wikipedia.org/wiki/Universally_unique_identifier
[7] https://spdx.github.io/spdx-spec/7-relationships-between-SPDX-elements/

2.3 Mapping to Existing Formats

Table 1 (recreated from Table 1 in "Survey of Existing SBOM Formats")[8] maps baseline component attributes across SPDX and SWID.

Baseline	**Representation in SPDX**	**Representation in 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:`	`<Link>@rel, @href`
SBOM Author	`(2.8) Creator`	`<Entity> @role (tagCreator), @name`

Table 1: Mapping baseline component information to existing formats

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 as necessary, an SBOM needs to capture supply chain relationships between components, to the extent that those relationships are known. While an SBOM may ultimately reflect many types of relationships, the baseline described in this document is built on a single type of relationship: "includes." That is to say, this component contains other sub-components, or dependencies. This captures the fact that the software code may be included in a component, even if there is no actual logical dependency (i.e., part of a library).[9]

An SBOM is made up of one or more components, and each component may also contain one or more components. The first component listed in an SBOM defines and identifies both the component and the current SBOM. We call this the primary component, and every SBOM has at least this one component.

[8] https://docs.google.com/document/d/1YzpLd8VtLIrAQtULZhRf6ZENqmaZ-w71wfhQdpNSpcs
[9] A richer set of relationships such as "patches to" or "expanded from" can be found in some of the existing standards. This document focuses on the single high level dimension of "includes."

In the simplest case of a root SBOM or component, consider a single component that was created entirely from scratch with no dependencies. The SBOM of this component consists of only one SBOM record. This single record defines the component and the SBOM and has a special relationship type of “self.”

In the example above, the SBOM of Acme Software would have four components, since each SBOM includes itself. We can visualize an SBOM as an upside down tree. The root of the tree is the software itself, and is not included in anything else. Each node is a component, and most components are made up of other components. The leaves of the tree do not include further subcomponents as part of this SBOM.

In this example, Acme makes Software that uses exactly two third party components, Bob’s Browser and Bingo Buffer. Bob’s browser, in turn, uses Carol’s CompressionEng, and may also use other components. Carol’s CompressionEng does not include further subcomponents, while Bingo Buffer may or may not have any further dependencies.

2.4.1 Knowledge About Relationships

Ideally, all suppliers will create and provide SBOMs for their components, and all consumers will obtain complete chains of these authoritative SBOMs. In every SBOM record, Author Name (2.2.1) will equal Supplier Name (2.2.2) and there will be perfect knowledge. Until this state of transcendent SBOM utopia is achieved, SBOM authors may want to make less-than-authoritative assertions about SBOMs for which the authors are not the suppliers. In such cases, it is important for authors to convey their knowledge about the relationships of other suppliers’ components.

The default presumption is that a component may have further unlisted subcomponents. When the author of an SBOM is confident that a component has no further subcomponents, or that they have fully enumerated all the subcomponents, that should be indicated. The following four assertions may cover the range of an author’s knowledge about another supplier’s components.

1. **No assertion**. This is the default. The component may or may not have any upstream relationships.
2. **None**. There are no upstream relationships, this is a root component.
3. **Some**. There is at least one upstream relationship and may or may not be others.
4. **Complete**. There is at least one upstream relationship and all upstream relationships are enumerated.

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, but they are conceptual representations and not format impementations like SPDX or SWID.

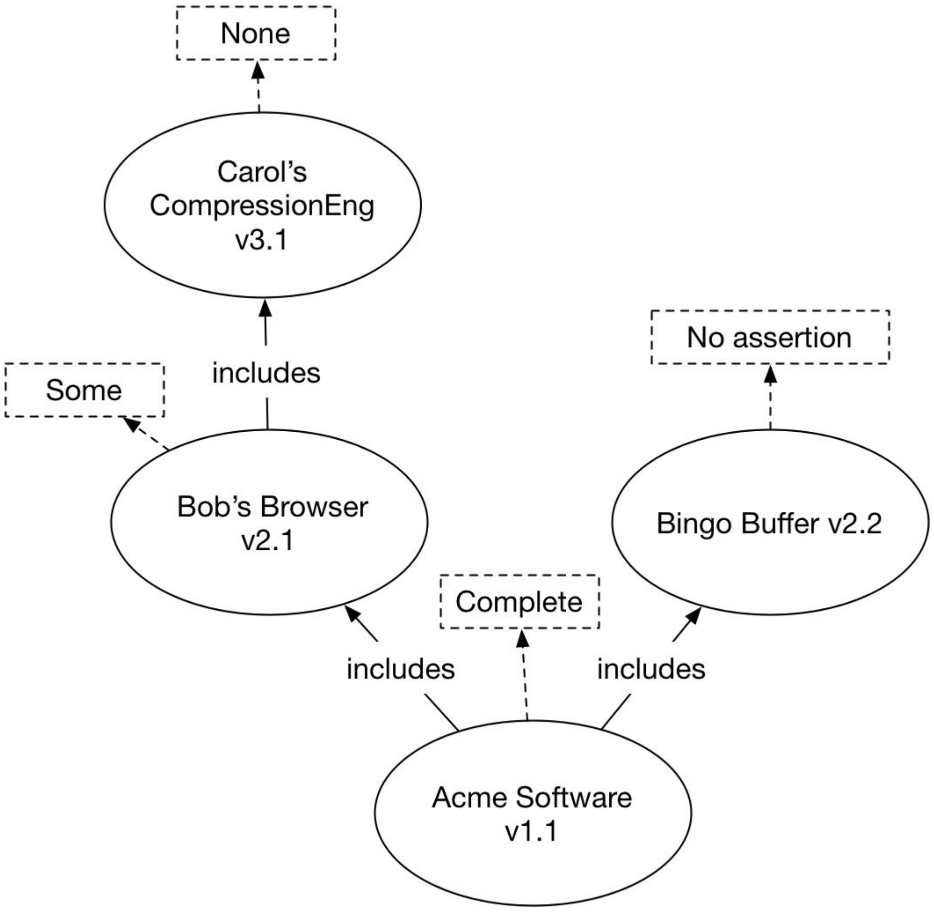

Figure 1: SBOM tree example

Relationship	**Component Name**	**Supplier Name**	**Version String**	**Author**	**Hash**	**UID**	**Relationship Assertion**
self	Acme Software	Acme	1.1	Acme	0x123	234	Complete
includes	|-- Bob's Browser	Bob	2.1	Bob	0x223	334	Some
includes	|-- CompressionEng	Carol	3.1	Acme	0x323	434	None
includes	|-- Bingo Buffer	Bingo	2.2	Acme	0x423	534	No assertion

Table 2: SBOM table representation for Acme Software

2.6 Additional Elements

Depending on the desired application or uses of an SBOM, further elements are likely necessary. The specific additional information needed will depend on the application and therefore each specific item should be considered for its value to expanding the potential use cases by adding it to the SBOM. In some cases, the optional elements may not add benefit for the planned use cases.

The following list describes several examples of additional information that could be useful to add to an SBOM:

1. **Intellectual property applications**, such as license management and entitlement, require 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.
2. High assurance applications may need information about the **pedigree** and **provenance of components** such as how they were built.
3. **End-of-life dates** and the ability to indicate what technologies a component supports would be useful for multiple applications.
4. **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 the user to identify all DNS-relevant components.

Figure 2 illustrates the relationship between baseline SBOM information and additional information required to enable different applications.

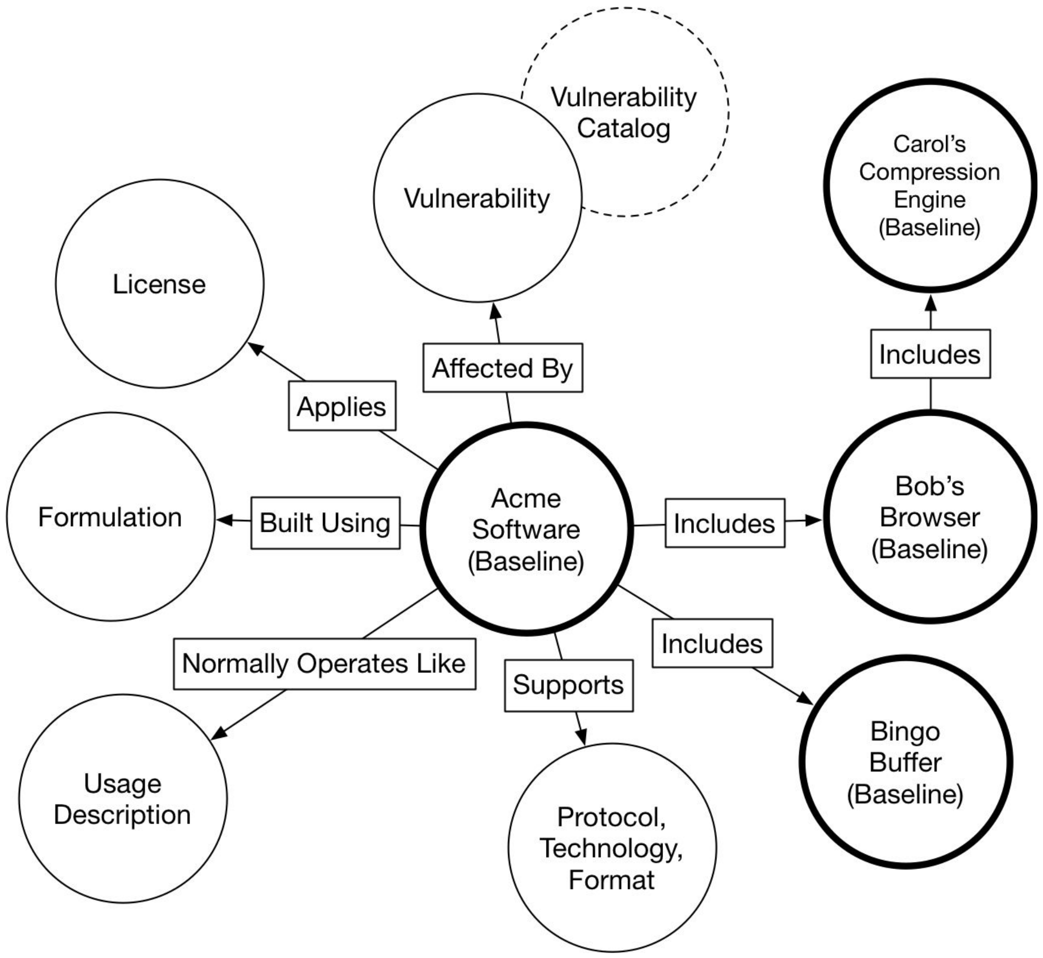

Figure 2: Example relationships between baseline and additional information

3 Data Formats

Much of the benefit of software transparency described in this document (and the accompanying Use Cases document) are easier to realize if SBOM data can be generated, shared, and processed by machine. This requires widespread interoperability across the supply chain which, in turn, requires standardized data formats.

Stakeholders formed a working group to explicitly set out to map existing standards that might be used. No single widely used standard was explicitly designed to only address SBOM use cases. However, 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. It became apparent that both standards can cover the basics of the baseline SBOM, and each have their own advantages and proponents; it seems unlikely in the near future that one will supplant the other across the entire software ecosystem.

Both formats focused on the core problem of identifying software entities and conveying associated metadata, and had the requisite fields to cover the needs for the baseline SBOM. Moreover, stakeholders were able to generate a mapping between SPDX and SWID covering the baseline SBOM. Thus, while different communities select the format that meets their needs, the baseline SBOM data can carry across the supply chain through straightforward translation. For more information about SPDX, SWID, and other existing SBOM formats, please see "Survey of Existing SBOM Formats."

4 SBOM Processes

This section describes SBOM use in business processes 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."[10]

The section highlights when and how an SBOM is created along with how it can be exchanged. Finally, a couple of real-world examples of applications are explored in detail here to understand specific scenarios like "License Management" and "Vulnerability Management" so you can see SBOM use as an independent data source as well as SBOM's applications to other datasets in typical business processes.

An SBOM, as depicted in section 2 of this document, includes much more descriptive information as well as aggregated information from other software as components. Each such components can also be an SBOM with one or more sub-components. SBOM then becomes a rich document that carries sufficient information to address many of the business process needs that involve producing, choosing or operating software. The SBOM's lifecycle (creation, updates and retirement) and exchange methods proposed here make it possible to support the business needs of a modern dynamic software driven industry.

4.1 SBOM Creation: How

To create an SBOM, the supplier assembles the baseline component information, optional attributes, and a list of any directly included components. The SBOM will then be assembled using tools constructed as integral to the supplier's software build process. Where SBOMs for included components are readily available, they can be included directly into the newly created SBOM. Where such information is not available, the supplier will provide "best efforts" 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..

Any entity creating, modifying, and packaging/delivering components is considered to be a supplier and is therefore responsible for creating SBOMs. This includes also system integrators who are essentially considered suppliers for SBOM purposes.

[10] https://docs.google.com/document/d/16hUeVaAuIFs6gbTkZ2u_34iOY_t0TDFKRi_IxF3gm0E

SBOM will include a number of defined attributes to identify components that were included in the software. Some of these attributes are mandatory while others are optional. In every case, SBOM serves as a system of record or authoritative source of information about software. However, some of this information may need to be validated with other external sources, for e.g., vulnerability information of a component can be derived from the National Vulnerability Database (NVD) using the component identified. The standards document also provides further details of these attributes in order to satisfy pressing business processes such as License Management and Vulnerability Management discussed here.

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 marked as updates or upgrades, releases of new versions, and patches.

A new SBOM should also be created when information about included components changes, even if the components themselves have not changed. For example, an upstream component is patched, or start providing SBOMs (thus removing the need for their downstream consumers to make SBOMs for themselves). In the former case, the underlying software is changed; in the latter, we have more information about the software, but still update the SBOM to pass this information along. This is essential for the SOM to serve as the authoritative source of the information on the current state of the software that will be packaged, delivered and used.

4.4 SBOM Exchange

It is necessary to exchange SBOM information. The primary exchange is directly from a supplier to a consumer with a single downstream supply chain hop. 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, OpenChain and ATOM 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.

4.5 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 many times 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 this tree (or graph[11]) structure, ultimate upstream or root suppliers only create original components and do not include components (have dependencies) from any other supplier. Components flow downstream from these root suppliers throughout the graph. At the bottom of the (upside down) tree, leaf consumers only obtain components and SBOMs and 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 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 authoritative information about their components and low costs to generate and share that information. 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 only acting as the author of the SBOM and not the creator of the software. 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

[11] The tree is likely a directed acyclic graph. Consider an upside down tree, root supplier nodes have no parents and leaf consumer nodes have no children.

be clear that they (the author of the SBOM) 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.

The concepts around SBOM exchange and network rules are designed so that consumers can obtain comprehensive lists of the components they use across different suppliers and supply chains. Figure 3 expands the example in Figure 1 to show a consumer connected to two different supply chains. The consumer in this example would have two SBOMs, one in Table 2 and one in Table 3.

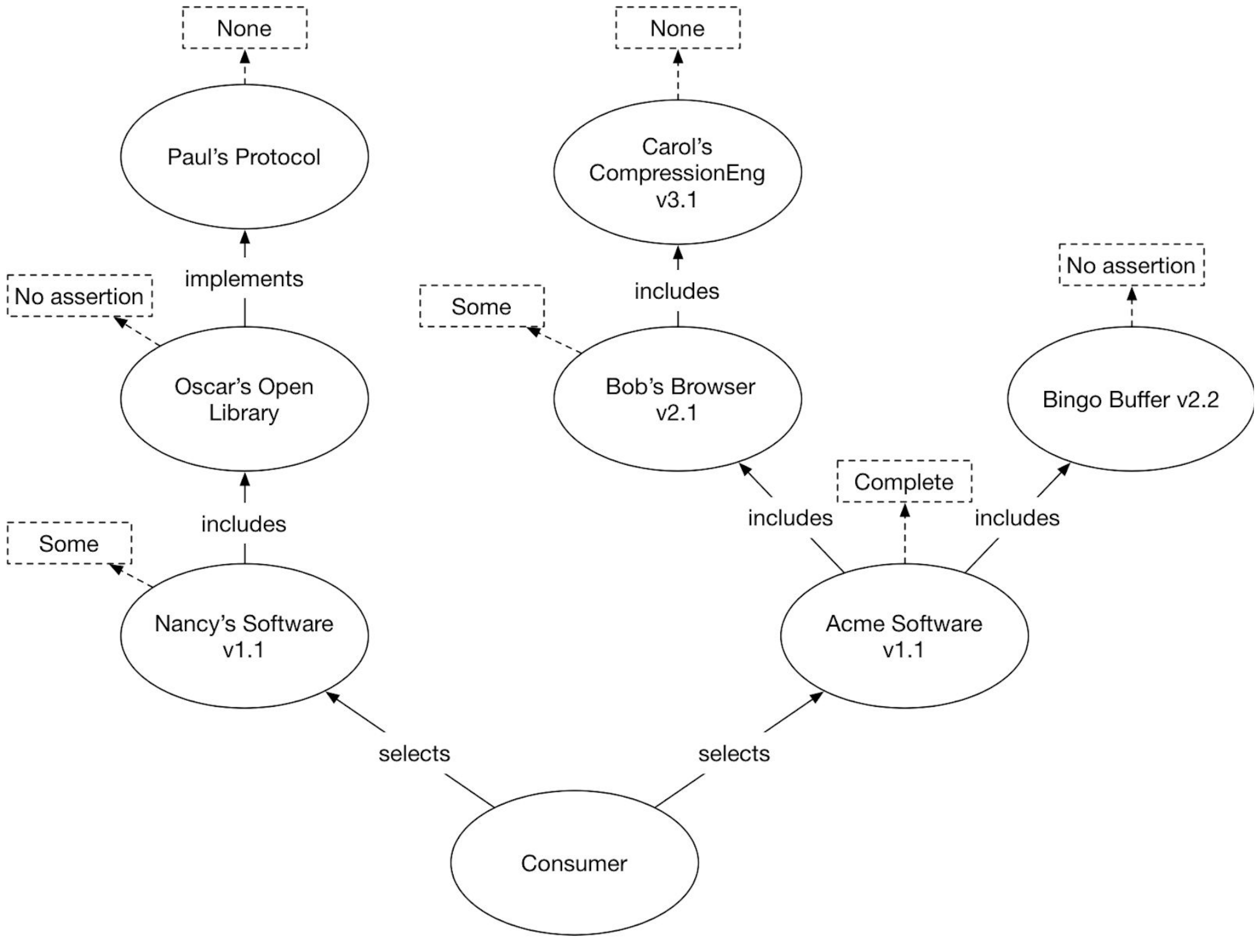

Figure 3: Consumer tree with two supply chains

Relationship	Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship Assertion
self	Nancy’s Software	Nancy	1.1	Nancy	0x523	237	Some
includes	|-- Oscar’s Open Library	Oscar	2.1	Nancy	0xA23	394	No assertion
implements	|-- Paul’s Protocol	Paul	2018	Nancy	0xB23	934	None

Table 3: SBOM table representation for Nancy’s Software

4.6 Roles and Perspectives

4.6.1 Perspectives

Different stakeholders will use SBOMs in complementary yet distinct ways. The “Roles and Benefits for SBOM Across the Supply Chain” document suggests three perspectives: those who produce, choose, and operate software.

4.6.1.1 Produce

Producers of software are typically, but not always, suppliers. A goal of this SBOM systems 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 another 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 3rd party components that were included in their products. Suppliers will surely benefit in many scenarios due to the clarity that will be brought by having an SBOM. In the case where a supplier does not provide detailed information of the sub-components there is a higher likelihood of lack of clarity causing the user to assume the worst about the unknown parts of the product.

Additional benefits to suppliers include the ability to determine which organization to contact to get fixes for published CVEs of 4th+ party components that have had recent fixes published for specific CVEs.

A key problem occurs when scanners run by a supplier or supplier's customers, indicate that CVE fixes are missing for Components in their distribution. The problem is that unless the Components are all directly included, the supplier does not know which directly included Components include the Sub-components missing CVE 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 for the CVE fixes.

Even better, if the SBOM (or associated information) contains information that the CVEs cannot be exploited via any of the directly included components, the product Supplier 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 supply them.

4.6.1.2 Chose

SBOMs can be used by prospective choosers (such as acquisition office) 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 information or license information.

Information such as licensing information, CVE related information or product lifecycle information can be readily used to determine if a prospective user will choose the product or not.

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. .

4.6.1.3 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.

4.7 Applications

The focus of the SBOM is on the core capability of identifying components and their relationships. Many use cases that rely on the baseline SBOM require additional information in order to function. In this section several major 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 archeological investigation to determine if a vulnerable sub-component is used, and if the vulnerability transitively makes the parent

component vulnerable. [cite wysopal] 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.

2. **Intellectual Property:** There are a number of intellectual property applications that could be improved with better inventory data. Managing software licensing -- 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.
3. **High Assurance:** High assurance of the source and integrity of components. Provenance and pedigree. Requires further information about suppliers, how components are built, the chain of custody as components move through the supply chain, how any modifications are made.
4. Better knowledge of components also supports supplier and supply selection.

See also “Roles and Benefits for SBOM Across the Supply Chain.”

4.8 Tool Support

Adoption is surely the key for SBOM to become effective and mature to meet these business needs and applications that are laid out through this section. Only a few suppliers and associated customers can justify the development of SBOM creating tools (e.g., swid-tools[12] and swidGenerator[13]). These tools can automatically generate SBOMs and can also collate related information on components and subcomponents.

To support the adoption of SBOM, tools to analyze and correlate SBOM information will be necessary. In many cases, build and development tools should be updated to produce SBOM information.

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 “drop-in” gramatical replacement for the term.

5.1 SBOM

(Software Bill of Materials)

[12] https://apps.fedoraproject.org/packages/swid-tools
[13] https://github.com/strongswan/swidGenerator

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 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 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.

5.3 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).

5.4 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.

5.5 Consumer

entity that obtains SBOMs

An entity can be both a supplier and consumer.

6 Background

6.1 Overview

On July 19, 2018, the National Telecommunication 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 by various government agencies that 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 a report produced by each of these groups. This report is the output of the “Understanding the Problem” task group and seeks to (1) describe the problems that are initiating the need for a software bill of materials (SBOM), (2) what an SBOM baseline document should include, and (3) an overview of supporting processes to manage SBOMs. The additional reports from the other task groups may be found at https://www.ntia.doc.gov/SoftwareTransparency.

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) identification of goals, problem statement, and scope, (3) useful terminology, and (4) 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.

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.
>
> (Source: https://www.ntia.doc.gov/SoftwareTransparency)

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 approach.

6.3 Scope

The scope of this initiative will include the definition of the structure of a Software Bill of Material (see Section 5 Terminology for definition), 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. While we do not specifically account for hardware, software doesn’t run by itself. A software system requires not only traditional computing hardware (e.g., CPUs, memory, disk, network, etc) but also 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 a harmonized 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.

7 Conclusion

Organizations across the globe face operational and supply chain related questions about whether the software they are using to do their day-to-day work and support their ongoing business areas contains known vulnerabilities. They are also trying to understand if software they are using is authentic and unaltered, its use is proper and legal, and they are approaching obsolescence or end-of-life issues with the software upon which they depend. As software is being used to run more and more of the critical aspects of organization’s business and systems these become increasingly pressing and unavoidable questions for almost everyone.

To be useful to end-user organizations an SBOM needs to include the basic information that enables correlating and connecting related information about the software as it moves through the supply chain and into operations. It is also critical that the SBOM be as small and concise as possible to help ease the integration and adoption and keep management of SBOM data as simple as possible. This is the reason that this effort sought to identify a most basic baseline SBOM, with only a few required attributes. As noted in this document, however, restricting an SBOM to only these baseline expectations will restrict the number of valid use cases achievable through the use of SBOMs. To be operationally useful any organization planning to utilize SBOMs for security of their systems will need to have activities that can connect the information in the SBOMs to their vulnerability management activities, but that mapping is a separate activity, one enabled and enhanced by future pervasive existence of SBOMs for all of the software in the enterprise.

While the use of SBOMs will not solve all security issues facing each of the industries involved in this initiative, it will help to increase transparency and better arm those defending their organizational networks and systems from new threats. As this space matures, the ready availability of SBOMs will hopefully then lead to further work in establishing more coordinated and standardized methods of sharing and managing the information stored in SBOMs. One of the reasons to standardize the structure and content of the SBOM is to allow for this next step. Tooling will likely be a major factor in the advancement in the area of SBOM management.

Overall, the goal is to ensure that the necessary information, as intended to be captured by the SBOM, is available to those who need it, thereby leading to better security management.

As software touches every area of our modern lives and is constantly evolving, SBOM si likely to take a similar path in becoming more applicable to our modern world. As SBOM applications and capabilities are exposed, various stakeholders are bound to find ways to extend it either using external data (such as vulnerability datate) or maturing what has been framed in this document. Adoption, tooling, and community efforts such as this one are expected to provide the bulk of this ongoing development to evolve SBOM to our modern software-driven world.

8 Acknowledgements

The chairs of the Framing Working group, Michelle Jump and Art Manion, thank the working group membership for their time and effort. Acknowledgement does not imply endorsement of the text.

Josh Corman
Christopher Gates, Velentium
Les Gray, Abbott
Ed Heierman, Abbott
JC Hertz
Kent Landfield
Eliot Lear
Bruce Lowenthal
Chujiao Ma
Bob Martin, MITRE Corporation
Chandan Nandakumaraiah, Juniper Networks
Vijay Sarvepalli
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.