Framing Software Component Transparency

NTIA Multistakeholder Process on Software Component Transparency
Framing Working Group
2019-06-25

1 Overview

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.

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.

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 systems and thus the root cause of incidents
- Reducing unplanned and unproductive work
- 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.

Scope

The scope of this initiative will include the definition of the structure of a Software Bill of Material (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 software should be considered in the scope of this initiative, including automotive, financial, healthcare, 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 (eg, 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. However, the group acknowledges that there are 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:

- Methodology for mapping vulnerabilities to components
- Stand-up of unified sharing mechanism for SBOMs
- Approach for documenting relationship between components for any given SBOM
- License management
- Lists of known vulnerabilities, exploitability, or patch level

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.

Goals

The overall goal of this initiative is to create a standardized model for software component information that is structured for sharing across multiple industry sectors, including how to create and share an SBOM.

This is intended to help reduce cybersecurity risk. It will be approached primarily by sharing information about software components with the necessary stakeholders so that users can do better risk management, specifically vulnerability management and impact assessments. With

this, the user can prioritize vulnerability responses and management, including adjusting the risk assessment and taking actions in addition to patching.

The information required is being set at a minimum needed for the purposes identified to minimize burden to vendors and users.

Process for Developing this Approach

This document, and accompanying related documents, were developed in an open, voluntary multistakeholder process convened by the National Telecommunications and Information Administration (NTIA). The first public meeting in July of 2018 allowed the participants in Washington and participating online to debate the overall issue of software component transparency. From this initial meeting, a series of working groups were established to tackle transparency from several different perspectives. One group tackled the overall challenge of what a “software bill of materials” should look like, the second documented existing and potential use cases for transparency, while the third reviewed the existing standards and formats that could be used to convey SBOM data. A final working group proposed and executed a “proof of concept” exercise with medical device manufacturers and the hospitals that used these devices.

Working groups met weekly or biweekly to define their charter and workstreams. All working groups were open, and new stakeholders joined as the project progressed. Each group was led by 2-3 co-chairs who volunteered from the community. The broader community reconvened every 3-4 months to share progress from the different working groups and discuss the overall direction of this initiative. NTIA acted as a convener and a neutral facilitator, and helping the different working groups coordinate as their work progressed. The actual documents were drafted by stakeholders with frequent opportunities for the community to weigh in as the deliverables took form.

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 data elements 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.

SBOM Elements

The Framing WG and the overall multistakeholder process reviewed existing software identification formats and thoroughly debated what elements are necessary to make 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 SBOM data. Without a way to systematically and consistently define and identify software components and their relationships, at scale, none of the desired applications are possible. Therefore, a baseline SBOM has been created, outlining several required elements, along with additional elements that are highly beneficial and recommended.

Required Elements

Baseline Component Information

The primary purpose of an SBOM record is to identify components. Some combination of the following baseline information is required to sufficiently uniquely identify components.

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

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.

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.

Version information about the 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].

Adding a cryptographic **hash** of the component is the most effective way to link a binary, as-built component to an SBOM. The hash is effectively the unique identifier of a component. Cryptographic signatures can be used in place of a hash, but add the complexities of key distribution and signature verification.

Relationships within and between SBOMs

A single flat SBOM is better than no SBOM, but to be truly effective and self-maintaining, an SBOM system needs to address supply chain relationships. In order to scale, an SBOM system needs a functional way to show relationships throughout the supply chain. We suggest the following design.

An SBOM is made up of one or more components, and each component is also an SBOM of 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.

Components are defined and identified using baseline component information as described previously.

In the simplest case of a root SBOM or component, consider a single component that was created entirely from scratch with no parent or upstream 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.”

Every other record listed below the primary component is a subcomponent of that component and has some type of upstream, parent, dependency relationship. These components are in some way used by or included in the current SBOM. This parent relationship can be further refined (e.g., derived_from, includes, as_known_as), but the sense of parent or upstream dependency is always maintained.

Every component is also an SBOM, so subcomponents listed in the current SBOM transitively includes all of their subcomponents, thus building the complete list of components.

SBOM as a Tree

Consider a global SBOM system as a tree (or hopefully a directed acyclic graph). Turn the tree upside down, so the roots are at the top and leaves at the bottom. Each node is a component, and most components are made up of other components.

Root components at the top of the upside down tree have no upstream dependencies. These components represent software built from scratch that includes no libraries or other components.

Leaf nodes at the ends of the upside down tree represent consumers of SBOMs that do not create components and do not provide SBOMs to anyone.

See graph in Figure N in the SBOM Example section at the end of this document. This subsection will be integrated more fully in future iterations of this document.

Sample SBOM

To help explain the way an SBOM is created using the required elements and built-in recursive relationships, consider the following examples using tables.

In this example, table N is the current or top-level SBOM. The first row (highlighted in blue) is the primary component that defines and identifies the current SBOM: the Acme Infusion Pump version 1.2.3. The pump contains three subcomponents. The Tomcat and Windows 10 SBOMs have been provided by their respective suppliers, Apache and Microsoft (note that the author

and supplier match). Acme has chosen to create their own SBOM entry for XMLSOFT Libxml2, presumably because XMLSOFT has not provided an SBOM (note that the author of the Libxml2 entry is Acme but the supplier is XMLSOFT).

Component Name	Supplier Name	Version String	Author	Hash
Acme Infusion Pump	Acme	1.2.3	Acme	xxxx
Tomcat	Apache	3.1.1.1	Apache	xxxx
Windows 10	Microsoft	1709	Microsoft	xxxx
Libxml2	XMLSOFT	1.8.10	Acme	xxxx

Table N: Current top-level SBOM for the Acme Infusion Pump

Table N is the SBOM for Apache Tomcat, which itself contains subcomponents. The first row identifies and defines this SBOM and the remaining rows are subcomponents.

Component Name	Supplier Name	Version String	Author	Hash
Tomcat	Apache	3.1.1.1	Apache	xxxx
Subcomponent 1	Name 1	1.2.3	Apache	xxxx
Subcomponent 2	Name 2	2.3.4	Name 2	xxxx
Subcomponent 3	Name 3	1.2.2.3	Name 3	xxxx

Table N: Subcomponent SBOM for Apache Tomcat

A consumer can connect multiple SBOMs to build a comprehensive list of all the nested components that are part of the current or top-level component.

For examples of SPDX and SWID data formats to support SBOM creation, please see SBOM Examples at the end of this document. This section will be further integrated into the document at a later date. Also see Section 3 Data Formats for additional context.

Optional Elements

Depending on the desired application or uses of an SBOM, further information is likely necessary to achieve the full benefit of the SBOM system.

For example, vulnerability management requires a **catalog of known vulnerabilities** (e.g., CVE), association of vulnerabilities to components, and possibly a means by which to convey the transitivity of a vulnerability from one component to another.

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.

High assurance applications would likely need information about the **provenance of components** and how they were built.

End-of-life dates and and the ability to indicate what technologies a component supports would be useful for multiple applications. Groups of components (around technologies or other concepts) could be implemented through the proposed recursive relationship model, treating a technology as an upstream component. For example, “component X implements technology Y” with component X including technology Y as a part of the component X SBOM.

Figure 1 illustrates the relationship between baseline SBOM information and additional information required to enable different applications.

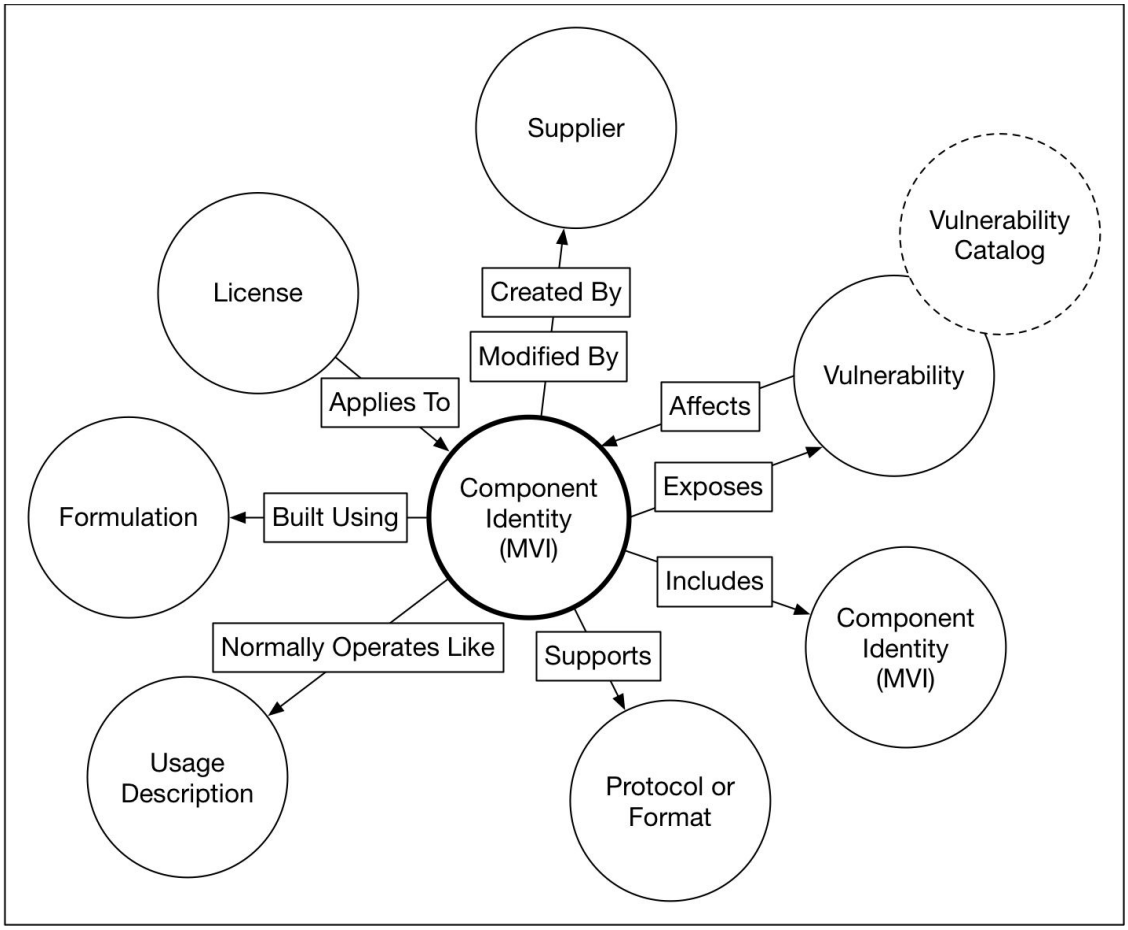

Figure 1: Relationships between baseline SBOM and additional information

Other Requirements

In addition to the baseline information, the SBOM system inherently supports (and requires) parent or dependency **relationships**. This is described above in [Relationships].

An SBOM system must also support the ability to cryptographically authenticate SBOM information. Creating signatures is optional. While not inherently part of the SBOM system, components themselves should be signed.

3 Data Formats

Much of the benefit of software transparency described in this document (and the accompanying Use Cases document) are much easier to realize if the data can be automatically generated and interpreted. This requires widespread interoperability across the supply chain which, in turn, requires a common set of data formats to convey the SBOM data.

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. It became apparent that both standards can cover the basics of the baseline SBOM, and each have their own advantages and proponents; it is unlikely that one will supplant the other across the entire software ecosystem.

Both of these 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, please see <https://docs.google.com/document/d/1YzpLd8VtLIrAQtULZhRf6ZENqmaZ-w71wfhQdpNSpcs/edit#heading=h.2a0qfwe12f7dt> that reviewed existing standards and spells out the basics of implementing SBOM in SPDX and SWID.

4 SBOM Processes

This section is still under construction. The content will need to be refined, edited, and structured in subsequent drafts. Please review the content and provide feedback on any of these subsections but recognize that this section is not yet finalized and fully edited.

This section describes SBOM use in business processes from four constituencies of users: Software creators, prospective software users, software users and tool suppliers. There are many business processes use cases that involve SBOMs and many of these are discussed in detail at [use cases doc?].

Some of these use cases involve SBOMs alone while others require mapping between SBOMs and data external to the SBOM. In this section, an example of a business process of License Management of software components will be used to illustrate use of SBOMs alone while an example of vulnerability management of Products will be used to illustrate use of SBOMs mapped to data external to the SBOM.

SBOMs from a Business Process Perspective

Software is built of components which are created and obtained from a variety of component creators, called suppliers. These Suppliers can combine root-components, which might be

transformed via helper tools or compilers, into aggregated components often combining and transforming already aggregated components at different levels into the resultant aggregated component, which is just called a component.

Such Components are often also called Products, in cases where they are offered to End Users, with the understanding that some End Users might use such “products” in constructing aggregated products of their own.

The SBOM system identifies components using baseline component information as described in section 2. The SBOM optionally contains additional information (beyond the baseline) about components and a list of included subcomponents. Each subcomponent is also an SBOM of one or more subcomponents.

For the software component license management example, the SBOM must include licensing information for components since many components can be used under different licenses. Thus, a mapping between baseline component information and license information for that component will only yield the list of possible licenses that can be selected for its use, not the license actually selected.

In the example case of vulnerability management, no information except the baseline component information is needed since that information can be used to map to vulnerability information published by the supplier or a vulnerability catalog such as CVE.

SBOM Creation: How

To create an SBOM, the product supplier assembles the baseline component information, optional attributes, and a list of any directly included components. The SBOM can then be assembled using tools constructed for that purpose. Where SBOMs for included components are available, they can be included in the directly included component list. Where they are not, the Supplier will provide “best efforts” SBOMs, which will be indicated by the fact that the Author for an included component SBOM will not match the Supplier in the baseline component information of that component.

There are a number of defined component attributes, some mandatory and some not. Examples are: the product licenses, license attributions, CVE related information and transformation information such as compilation transformations or helper tool transformations.

In the product license use case, the license selected for the use of each component must be included as an attribute since there is no other way to get this information except in an SBOM structure.

In the vulnerability management use case example, no additional information need be provided for a component beyond the baseline component information which can be used to map to the NVD published defect information and to the Supplier site that provides defect remediation

information. This assumes there is a mapping available between the baseline component information and published vulnerability information.

However, some additional information might be included such as a list of CVEs pertaining to included components that cannot be exploited in the context of components including the CVE associated component. This information also might be provided at a Suppliers site.

SBOM Creation: When

SBOMs should be created or updated when components are created or updated. Changes to components require corresponding changes to SBOMs. Changes to components are often marked as updates or upgrades, releases of new versions, and patches.

SBOMs should also be updated when information about included components changes, even if the components themselves have not changed. For example, an upstream component might change its license, or start providing SBOMs (thus removing the need for their downstream consumers to make SBOMs for themselves).

SBOM Exchange

It is necessary to exchange SBOM information. The primary exchange is from a supplier to a user directly one supply chain hop away. 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 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 unique ID to look up SBOM information on a supplier’s website. Problems here with link rot. Other techniques exist including MUD, Atom, ROLIE, OpenChain.

Want tools to consume and store/process/analyze SBOM data. Installers/package managers, asset management/SAM, stand-alone tools that can collect SBOM files/content.

Network Rules

Participants in an SBOM system act as suppliers creating SBOM information, consumers receiving it, or in most cases, both. Participants follow these network rules.

Suppliers define components. An SBOM contains components that can be

1. Initially created from scratch by a supplier
2. Created by the component's supplier (who also creates and provides the SBOM)
3. Created to identify a component from a different supplier (who does not create and provide SBOM for their component)

Suppliers create SBOMs for the components they define. As described in [recursive relationships], an SBOM must list at least one component, which is also identifies the SBOM.

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.

Root suppliers only create original components and do not include components from any other supplier. Leaf consumers only obtain components and SBOMs and do not produce components or create SBOMs. Most participants act as both suppliers and consumers.

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 components and low costs to generate and share that information. This model also distributes the cost to produce SBOM information proportionally (fairly?) across suppliers.

Most participants of an SBOM system act as both suppliers and consumers. Even end-user organizations may act as suppliers, producing SBOMs for in-house components or external components such mobile applications or devices.

A supplier may create SBOMs for third-party components that the supplier does not create or define themselves. This is typically necessary when an upstream or parent supplier is not providing authoritative SBOMs. 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 should 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.

Use Cases and Applications

Use Cases

Different stakeholders will use SBOMs. The Use Cases WG [doc] suggests three perspectives: those who produce, choose, and operate software.

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 will 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 included in their products. These include the benefits noted above but in addition include the benefit of determining 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 a 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 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.

Chose

SBOMs can be used by prospective users considering the use of a component or product that has an associated SBOM. The users might be interested in information directly attributable to the product, such as its baseline component information or license information. They might also be interested in information of components included in a product distribution.

Information such as licensing information, CVE related information or product lifecycle information might be used to determine if a prospective user will use the product or not. Since all CVE or product lifecycle related information may not be in the SBOM or included SBOMs themselves, the prospective user may need to consult other information sources such as NVD or the Component Supplier's site for lifecycle related information such as end of support dates, evidence of lack of support or available fixes for outstanding CVEs.

In the license management use case, the prospective user views the license of every constituent component for a Product under consideration for use in addition to the Product itself. These licenses can then be considered for suitability for use by organizations that might reject a component requiring royalty payment, one only obtainable under the GNU license or for other reasons. As many components, such as MySQL database have multiple licenses, such information cannot be obtained from sources external to an SBOM.

Since licensing issues are typically only considered during product version selection, the license management use case will not be discussed in the following sections except the tools section.

In the defect management use case, the prospective user may use the baseline component information to map to an NVD history of published vulnerabilities of a product or its constituent components and to the Supplier support site to determine the frequency of issuing patches, time between publishing patches for third party components and their inclusion in a product or that the product or its components apparently are no longer supported. A prospective user might reject product use because fixes are issued too frequently; because fix inclusion of third party component published fixes is delayed excessively; or for other reasons.

Operate

For current users of a product associated by an SBOM, there are a number of SBOM uses. Some of this information may be static, such as licensing information. Some information may change or be updated after a product's initial distribution.

Most of the information of ongoing interest for End Users is expected to be found in SBOM updates, or associated information such as NVD CVE information, availability of new versions, availability of new features or end of life notifications.

Such information might be used, for example, in cases where a product customer wants to "Trust but verify" that the Supplier has included published fixes into the product version the customer is using.

Applications

We focus on the core capability of identifying components and their relationships. Many use cases that rely on this core MVI SBOM require additional information in order to function. Here are three prominent applications.

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. Can't defend what you don't know you have.

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 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.

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.

Better knowledge of components also supports supplier and supply selection. See the Practices WG document for more. And use term from Practices WG document in this section as appropriate.

https://docs.google.com/document/d/16hUeVaAuIFs6gbTkZ2u_34iOY_t0TDFKRi_IxF3gm0E

		Application			
		Intellectual property	Vulnerability management	High assurance	Other...
Use case	Produce software				
	Choose software				
	Operate software				

Table N: Use case perspectives and applications

Tool Support

Few of the individual product Suppliers and associated customers can justify the creation of SBOM creating tools, tools that provide reports of SBOM contents, and/or tools that use SBOMs in conjunction with other sources of related information such CVE, end of life, or component Supplier contact information.

Increased use of SBOMs provides increased opportunities for tool vendors to create or update tools that fill the needs of SBOM creation and analysis. As such tools become available and are improved, their availability will increase the demand for SBOM use which will increase the demand for better tools, etc.

Availability of quality SBOM tools will allow SBOM creation and use as well as accelerate business processes that are dependent on information that can be derived from SBOMs and associated information sources.

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.

SBOM

(Software Bill of Materials)

list of one or more identified components and other associated information

Note: 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. Hardware is not excluded, but not the primary focus.

Component

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

Note: 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 like a .rpm. In SPDX terms package, file, and snippet map to “component.” In SWID terms… Source is not excluded, but is not the primary focus.

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).

Supplier

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

Note: A supplier may also be known as a manufacturer, vendor, developer, maintainer, or provider. Ideally, all suppliers are also authors of SBOMs for the suppliers’ components.

Consumer

entity that obtains SBOMs

Note: An entity, in fact most suppliers, can be both a supplier and consumer.

Example SBOM Formats

This section provides examples of ways to record SBOM information. See also the S&F document

https://www.ntia.doc.gov/files/ntia/publications/ntia_SBOM_standards_whitepaper_april11_draft.pdf

Basic Table

While not recommended for reasons of scalability and machine readability, it is possible to record SBOM information in a basic table, such as a spreadsheet.

	A	B	C	D	E	F
1	**Supplier**	**Component**	**Version**	**Hash**	**Includes**	
2	OpenSSL	OpenSSL	0.9.8a	0x113a8...	N/A	
3	Apache	httpd	1.3.26	0x33af2...	OpenSSL 0.9.8a	
4	MDM1	FooPump	4.0	0x44a83...	Apache httpd 1.3.26	

Figure N: Basic table SBOM example

namespace:name

While high level…

```
org.openssl:”OpenSSL 0.9.8a”
org.apache:”httpd 1.3.26”
com.mdm1:”FooPump 4.0 0x44a83…”
```

Figure N: Generic namespace:name example

package URL (purl)

Words about purl. The “requires” qualifier was created for this example and is type-specific to the “device” and “tgz” types.

https://github.com/package-url/purl-spec

```
pkg:tgz/org.openssl/OpenSSL@0.9.8a

pkg:tgz/org.apache/httpd@1.3.26?requires=pkg:tgz/org.openssl/OpenSSL@0.9.8a

pkg:device/com.mdm1/FooPump@4.0?hash=0x44a83…&requires=
pkg:tgz/org.apache/httpd@1.3.26
```

Figure N: purl example

SWID

Words about SWID.

```
<SoftwareIdentity name="openssl" tagId="openssl/openssl@0.9.8a" version="0.9.8a"/>

<SoftwareIdentity name="apache_httpd" tagId="apache/httpd@1.3.26" version="1.3.26"/>
<Link href="swid:openssl/openssl@0.9.8a" rel="requires"/>

<SoftwareIdentity name="MDM1 FooPump" tagId="MDM1/FooPump@4.0" version="4.0"/>
<Link href="swid:apache/httpd@1.3.26" rel="requires"/>
```

Figure N: SWID example

SPDX

Words about SPDX.

```
PackageName: openssl
SPDXID: openssl/openssl@0.9.8a
PackageVersion: 0.9.8a

PackageName: apache_httpd
SPDXID: apache/httpd@1.3.26
PackageVersion: 1.3.26
Relationship: openssl/openssl@0.9.8a PREREQUISITE_OF apache/httpd@1.3.26
```

```
PackageName: ”MDM1 FooPump”
SPDXID: mdm1/foopump@4.0
PackageVersion: 4.0
Relationship: apache/httpd@1.3.26 PREREQUISITE_OF mdm1/foopump@4.0
```

Figure N: SPDX example

Graph

While not an example of an SBOM format, the right side of figure N illustrates the relationships between components as described in the previous examples.

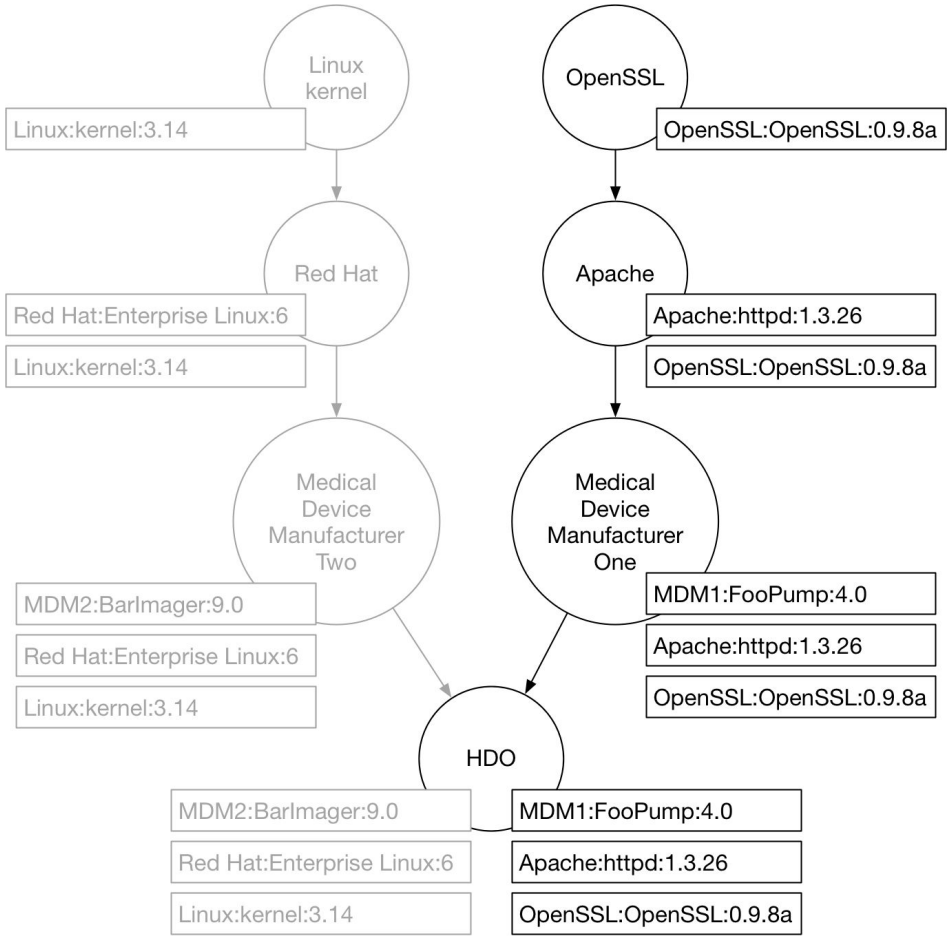

Figure N: Graph example