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

Second Edition

NTIA Multistakeholder Process on Software Component Transparency
Framing Working Group
2021-10-21

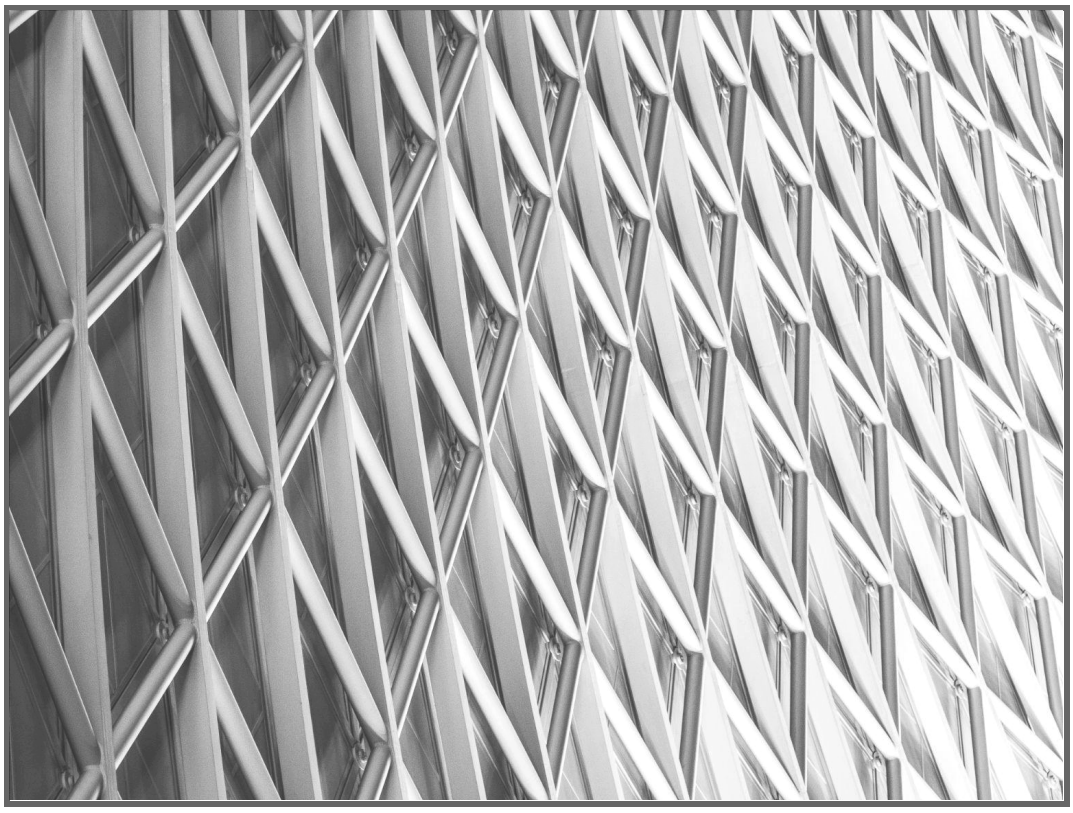

Photo by Bruno van der Kraan on Unsplash

Table of Contents

Foreword

The first version of this document was published in 2019 as part of the Phase I series of reports from the NTIA Software Component Transparency multistakeholder process.[1] The concept and implementation of SBOM is practicable today, but it will continue to evolve. This updated document reflects the work done by the NTIA multistakeholder process since the original publication. The update focuses on specific topics, not a comprehensive revision of the entire document. The updates chosen for this revision were based on insights by the Framing group since the initial report release as well as feedback from other task groups within the NTIA Software Transparency group, and experience from implementation in the broader SBOM community. These changes are highlighted in Section 7.

[1] https://ntia.gov/files/ntia/publications/framingsbom_20191112.pdf

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.[2] 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 components. So a subsidiary goal was to select a core, baseline set of attributes necessary to identify components with sufficient relative

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

uniqueness. Another goal was to capture SBOM applications and consider what additional, optional attributes and external elements might be needed beyond the baseline set.

Further background on this multistakeholder process and the Framing working group can be found in Section 5.

2 What is an SBOM?

An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships. These inventories should be comprehensive – or should explicitly state where they could not be. SBOMs may include open source or proprietary software and can be widely available or access-restricted.

SBOMs should also include baseline attributes with the ability to uniquely identify individual components in a standard data format. The most efficient generation of SBOMs is as a byproduct of a modern development process. For older software, less-automated methods exist.

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.

Defining baseline attributes (Section 2.2) and processes (Section 3) allows adoption by a variety of stakeholders quickly, which can 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 provide significant value 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., Common Vulnerabilities and Exposures, CVE[3]), associations of

[3] https://www.cve.org/

vulnerabilities to components (e.g., the use of Common Platform Enumeration, CPE[4] in the U.S. National Vulnerability Database, NVD[5]), and possibly a means by which to convey the exploitability or exposure of a vulnerability at different points along supply chains. The use of SBOM for license management requires that licenses and their restrictions are mapped to components.

> A note about terminology in this document: *Components* are units of software defined by suppliers or authors. *Attributes* are information about components and SBOM entries, primarily designed to identify components. An *SBOM entry* identifies a component and its associated attributes. An *SBOM* is a collection of one or more *SBOM entries*.
>
> Software that might commonly be called a "product" is treated as a type of component, often considered to be the primary component and subject of the SBOM.
>
> More terms are defined in Section 4.

2.1 SBOM Elements

The NTIA Software Component Transparency multistakeholder process reviewed existing software identification formats, considered feedback from the Healthcare Proof of Concept exercises, and thoroughly debated and questioned which elements would be necessary to create a scalable and functional SBOM system. Many of the answers depend on the desired use cases that can be built on top of sufficient quantity 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 use cases will function at scale. Section 2 proposes and describes the minimum elements and baseline attributes (Section 2.2) needed for any SBOM use cases.

2.2 Baseline Attributes

The primary purpose of an SBOM is to uniquely and unambiguously identify software components and their relationships to one another. Therefore, one necessary element of an SBOM system is a set of baseline attributes that can be used to identify components and their relationships. An SBOM system for format that follows the guidance and framing proposed in this document *must* support these baseline attributes. An SBOM system or format *may* support additional attributes. There are also cases where certain attributes may not be available or applicable or may not materially contribute to component identification (see Section 2.3).

[4] https://nvd.nist.gov/products/cpe
[5] https://nvd.nist.gov

The Author Name and Timestamp attributes provide meta-information about an SBOM; the remaining attributes apply to components. See *Software Identification Challenges and Guidance* for a more detailed examination of component and supplier identification.[6] A subset of these attributes (except component hash) are listed as necessary Data Field elements in *The Minimum Elements For a Software Bill of Materials (SBOM)*.[7]

2.2.1 Author Name

author of the SBOM

The author may not always be the supplier, which indicates that the SBOM was not created by the supplier.

2.2.2 Timestamp

date and time when the SBOM was last updated

Timestamp should be updated when an SBOM entry is changed, which includes initial creation. Timestamp should be consistent across time zones and locales and use a common international format, such as ISO 8601.[8]

2.2.3 Supplier Name

name or other identifier of the supplier of a component in an SBOM entry

Supplier Name should include some capability to handle multiple names or aliases. When the author and supplier are the same, the supplier created an SBOM for their own component. When the author and supplier are different, the author is making claims or assertions about a component for which the author is not the supplier. Supplier identification is not further specified in this document, but may be a significant contributor to achieving component identification at scale.[9]

2.2.4 Component Name

name or other identifier of a component

Component Name should include some capability to handle multiple names or aliases. Suppliers (or authors) define components and their names.

---

[6] https://www.ntia.gov/files/ntia/publications/ntia_sbom_software_identity-2021mar30.pdf
[7] https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom
[8] https://www.iso.org/iso-8601-date-and-time-format.html
[9] See Section 5 of https://www.ntia.gov/files/ntia/publications/ntia_sbom_software_identity-2021mar30.pdf

Component names can convey supplier names. Component (and Supplier) Names can also be conveyed using a generic **namespace:name** construct. One consideration is to use the Supplier Name as the namespace designator.

2.2.5 Version String

version of a component

Version information helps to further identify a component. Suppliers and Authors are free to choose a versioning scheme; one suggestion is Semantic Versioning.[10]

If a component does not initially have a version string, authors should create one.

2.2.6 Component Hash

cryptographic hash of a component

A cryptographic hash is an intrinsic identifier for a software component.[11] Digital signatures can be used in place of hashes and provide stronger integrity and authenticity, but add complexities including key management and signature verification. In addition to hash values, it must be clear to SBOM consumers as to how the hash was generated so that it can be reproduced. For example, the Software Package Data Exchange (SPDX) format specifies a Package Verification Code that creates a hash of hashes for individual file components.[12] SBOM formats are responsible for specifying how to produce hashes.

It is possible, and may be beneficial, to provide multiple hashes for a component or collections of components. Suppliers and authors choose how to define components, which in turn defines the scope of the hash. For example, an SBOM could include a hash for a source component, a hash for the compiled binary form of that component, and a hash for a collection of components.

As noted previously, Component Hash is a recommended, but not required, Data Field element in *The Minimum Elements For a Software Bill of Materials (SBOM)*.[13]

2.2.7 Unique Identifier

additional information to help uniquely define a component

A unique identifier can be generated relative to some globally unique hierarchy or namespace or reference an existing global coordinate system. Some systems that could be used as unique

[10] https://semver.org
[11] https://hal.archives-ouvertes.fr/hal-01865790/file/main.pdf
[12] https://spdx.org/spdx-specification-21-web-version#h.2p2csry
[13] https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom

identifiers include Common Platform Enumeration (CPE),[14] Package URL (PURL),[15] Universal Unique Identifier (UUID) (also known as Globally Unique Identifier [GUID]),[16] and Software Heritage ID (SWHID).[17] Component Hash (Section 2.2.6) may effectively function as a Unique Identifier.

2.2.8 Relationship

association between SBOM components

The relationship between components is inherent in the design of the SBOM model. The default relationship type is *includes.* This represents the inclusion of or dependency on a separate upstream component. To simplify presentation, this document reverses the direction of the relationship to *included in*. The choice of direction is not important to the model, as long as one direction is chosen and used consistently. Using the example from Section 2.6, 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.

A relationship type of *primary* is used when a component has no upstream dependencies identified in the SBOM. The primary component defines the subject of the SBOM (e.g., Acme Application in Table 2), including cases where the SBOM only includes one component (e.g., Carol's Compression Engine in Table 3).

Other types of relationships are discussed briefly in Section 2.5.The relationship attribute also allows an SBOM author to express their knowledge of additional upstream relationships and SBOM information, see Section 2.5.1 for more information.

2.3 Undetermined Attribute Values

There are cases where certain attributes may not be available, may not make sense, or may not materially contribute to component identification. One significant contributing factor is the lack of first-hand knowledge about the composition of components. When the author of the SBOM is not the supplier of the software component, the author may lack information or visibility necessary to generate some attributes. Another factor is the point in time at which the SBOM (and the component) are created, roughly: pre-build, at build or packaging time, and post-build. For example, binary software composition analysis performed post-build by a (non-supplier) author may detect a component but not extract the binary component to generate a hash.

[14] https://nvd.nist.gov/products/cpe
[15] https://github.com/package-url/purl-spec
[16] https://en.wikipedia.org/wiki/Universally_unique_identifier
[17] https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html

SBOMs must gracefully handle cases of missing or non-applicable attributes. A basic recommendation is to always provide all of the baseline attributes but explicitly define values that differentiate between “no assertion” (i.e., data is missing), and “no value” (i.e., the attribute is not applicable for this specific SBOM). Alternatively, an SBOM format can permit missing baseline attributes and treat them as default values (i.e., “no assertion” or “no value”).

2.4 Mapping to Existing Formats

Table 1 (recreated from Table 1 in *Survey of Existing SBOM Formats*[18]) maps baseline attributes across SPDX, CycloneDX, and SWID. More formats and their mappings are described later in the *Survey* document. In addition to the baseline attributes, authors should conform to the specifications of their chosen SBOM formats.

Attribute	**SPDX**	**CycloneDX**	**SWID**
Author Name	(2.8) Creator:	metadata/authors/author	<Entity> @role (tagCreator), @name
Timestamp	(2.9) Created:	metadata/timestamp	<Meta>
Supplier Name	(3.5) PackageSupplier:	Supplier publisher	<Entity> @role (softwareCreator/publisher), @name
Component Name	(3.1) PackageName:	name	<softwareIdentity> @name
Version String	(3.3) PackageVersion:	version	<softwareIdentity> @version
Component Hash	(3.10) PackageChecksum: (3.9) PackageVerificationCode:	Hash “alg”	<Payload>/../<File> @[hash-algorithm]:hash
Unique Identifier	(2.5)SPDX Document Namespace (3.2) SPDXID:	bom/serialNumber component/bom-ref	<softwareIdentity> @tagID
Relationship	(7.1) Relationship: DESCRIBES CONTAINS	(Inherent in nested assembly/subassembly and/or dependency graphs)	<Link> @rel, @href

Table 1: Mapping baseline component information to existing formats

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

2.5 Component Relationships

An initial step in developing an SBOM is to enumerate first-level components that a supplier directly includes in the primary component. However, in order to scale effectively, an SBOM needs to capture nested supply chain relationships between components, to the extent that these relationships are known. Bills of materials for physical components often describe these relationships as a "Multi-level BOM."[19]

While an SBOM may support many types of relationships, the baseline Relationship attribute described in Section 2.2.8 defines a single type of relationship: *includes* (or *included in*). An upstream component (often called a dependency) is *included in* a downstream or component. In Figure 1, Bingo Buffer is an upstream dependency of Acme Application, Bingo is an upstream supplier to Acme.

Other types of relationships may be necessary or useful, and existing SBOM formats support different types of relationships. It is possible to further refine the *included in* relationship, for example, conveying the difference between

- directly including, unchanged, an upstream binary component;
- including an upstream source code component, unchanged, by linking or compiling; and
- selecting an upstream source code component, modifying (forking) it, then including it by linking or compiling.

Modifying a component effectively creates a new component (e.g., a fork) and the modifier becomes the supplier for that new component. It is important in this example to maintain the heritage of the modified component and convey that it has been modified. For example, SPDX supports GENERATED_FROM and DESCENDANT_OF[20] relationship types.

While upstream components are typically included to provide functionality, it is common for parts of the component to not be used. A software program (component) might include a library (component), but only call some of the functions provided by the library. Or, certain features of a component may be disabled during build or packaging. This becomes important in some SBOM use cases, particularly vulnerability management. If a vulnerability affects an upstream component, the vulnerability may or may not affect downstream components.[21] Vulnerability Exploitability eXchange (VEX) is designed to convey the status of vulnerabilities in components.[22]

Component relationships are illustrated in Section 2.6.

---

[19] https://medium.com/@openbom/openbom-fundamentals-all-about-openbom-multi-level-boms-f06f50ca7f74
[20] https://spdx.github.io/spdx-spec/7-relationships-between-SPDX-elements/
[21] https://www.ntia.doc.gov/files/ntia/publications/wysopal_swct_kickoff_perspective.pdf
[22] https://ntia.gov/files/ntia/publications/vex_one-page_summary.pdf

2.5.1 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 (2.2.3), 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 components 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.

These relationship assertions can be recorded using an additional, optional 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.[23]
2. **None**. There are no immediate upstream relationships. As defined by the supplier, the component has no upstream components.
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 is 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 components 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. While both interpretations can be useful, it is important to recognize the limited scope of the first.

2.6 SBOM Examples

To further illustrate the relationships described in the previous section, consider these SBOM examples. Figure 1 and Table 2 show two different approaches to viewing SBOM information

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

and relationships. These are conceptual representations and not specific formats like SPDX, CycloneDX, or SWID.

In Figure 1 and Table 2, the SBOM authored by Acme for the component named “Acme Application” has four components. One of these, the primary component, is the Acme Application, which defines the subject of the SBOM. Acme makes a component named “Application” that uses two upstream components, Bob’s Browser and Bingo Buffer. In this example, Acme was able to obtain SBOM information from Bob about Bob’s Browser, which, in turn, uses Carol’s Compression Engine and possibly other upstream components. Acme was not able to obtain SBOMs from Carol or Bingo, so Acme authored SBOMs for those components. Carol’s Compression Engine does not include upstream components, while Bingo Buffer may or may not have any upstream components.

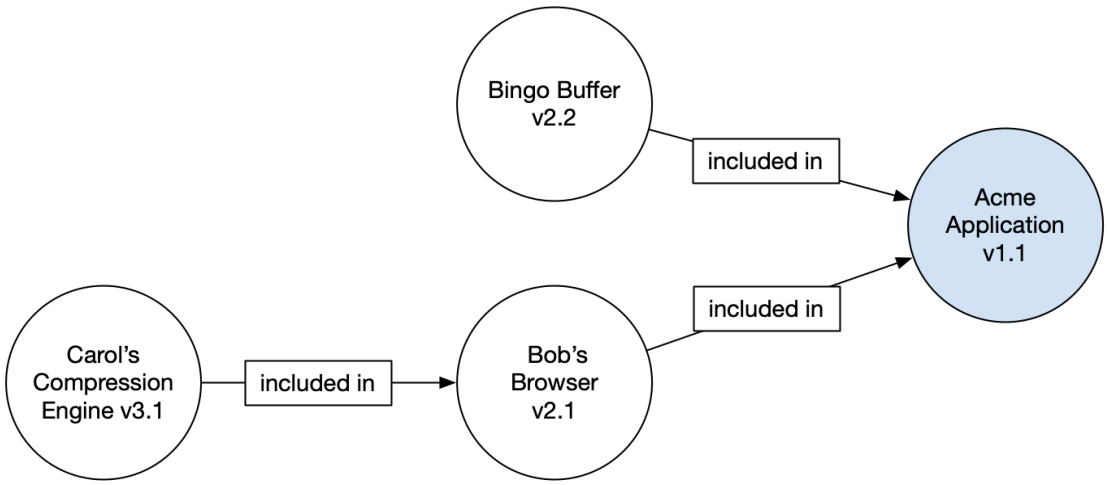

Figure 1: Conceptual SBOM graph

Component Name	Supplier	Version	Author	Hash	UID	Relationship
Application	Acme	1.1	Acme	0x123	234	Primary
|--- 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[24]

[24] In all similar tables, the Timestamp attribute is omitted and other attribute names shortened for presentation purposes.

In the simplest case, a single component is created entirely from scratch with no dependencies: Carol's Compression Engine v3.1. The SBOM for this component consists of only one entry that defines both the component and the SBOM using the relationship type of "primary." 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	Primary	None

Table 3: Conceptual SBOM table for a single (and primary) component

In Figure 2 and Table 4, the SBOM authored by Acme for the component “Acme Application” also has four components, and the relationship information is now enhanced with assertions regarding completeness.

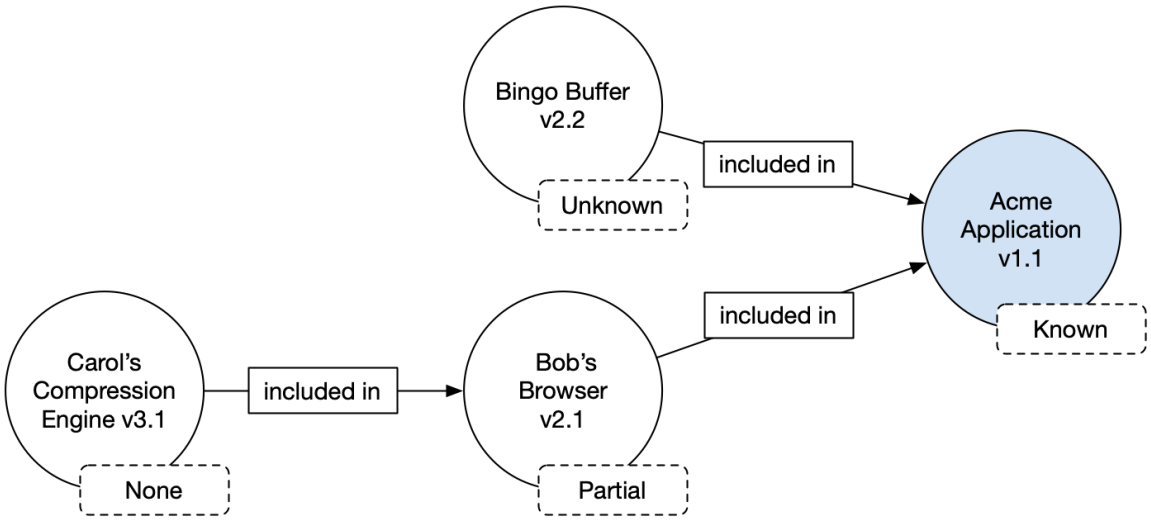

Figure 2: Conceptual SBOM graph with upstream relationship assertions

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Assertion
Application	Acme	1.1	Acme	0x123	234	Primary	Known
|--- Browser	Bob	2.1	Bob	0x223	334	Included in	Partial
|--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in	None
|--- Buffer	Bingo	2.2	Acme	0x423	534	Included in	Unknown

Table 4: Conceptual SBOM table with upstream relationship assertions

Acme Application (the subject and Primary component of this SBOM) asserts *Known* since all immediate upstream dependencies are covered. Bob’s Browser asserts *Partial* since at least Carol’s Compression Engine is upstream of it. Carol’s Compression Engine has no upstream components and the relationship assertion is *None*. Bingo Buffer is known to be an immediate upstream dependency of Acme Application, but since nothing is known upstream of Bingo Buffer, the relationship assertion is *Unknown*.

2.7 Additional Elements

In addition to baseline attributes, an SBOM likely requires additional elements and component attributes in order to support different use cases. The specific information needed depends on the use case, and not all additional elements or attributes will support each use case. Section 3.6 describes SBOM use cases, and some of the additional elements and attributes needed for those use cases. Additional elements and attributes include but are not limited to:

- End-of-life or end-of-support dates for components
- The ability to indicate what technologies a component implements or supports
- Mechanisms to group components, possibly by product lines or implemented technologies (a group could be treated as a special type of upstream component)

For example, knowing that “component X and component Y implement DNS” allows a user to identify all DNS-related components and treat them as a collection.

2.7.1 Authenticity and Integrity

An SBOM ecosystem must support the ability to cryptographically authenticate and verify SBOM information. In general, this means that authors must be able to digitally sign SBOMs and consumers must be able to verify signatures. Authentication and integrity protection requires appropriate digital signature and public key infrastructure.

3 SBOM Processes

This section describes how to create and exchange SBOM information 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*.[25] Three SBOM use cases—vulnerability management, intellectual property, and high assurance—are discussed briefly, to illustrate SBOM as an independent data source as well as how SBOMs can be integrated into typical business processes.

3.1 SBOM Creation: How

To create an SBOM, the supplier defines components that the supplier creates themselves, produces baseline and any additional attributes for those components, and enumerates 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 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 incorporated into 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 2.5.1 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 essential, 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).

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

3.2 SBOM Creation: When

An SBOM should be created when a new component is released. This loosely corresponds to build, packaging, or deployment activities. The SBOM should be updated when the component changes, including when new upstream components are added. The SBOM should also change when new SBOM information becomes available even if the components themselves have not changed. Changes to components are often noted as updates, upgrades, releases, and patches. Ideally, changes to components are indicated by a change in the Version String attribute. Maintaining current SBOM information is essential.

When changing an existing component (including patching or updating), 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

3.3 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 consumer 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, one option is to provide a URL 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. The Internet Engineering Task Force has developed a protocol and format for end user discovery of SBOMs, whether they are shared on a local device or on a web site. The specification is "format neutral," meaning it can support SPDX, CycloneDX, SWID, and future

formats as well.[26] Protocols such as ROLIE,[27] OpenChain,[28] MUD,[29] and ATOM[30] can be leveraged to exchange large amounts of information, yet have the ability to collect it on-demand and in an efficient manner. The availability of SBOM using some of these dynamic protocols will be a key to continued adoption of the SBOM.

3.4 Network Rules

Participants in an SBOM system include suppliers and authors creating SBOM information and consumers receiving it, as well as providers of optional intermediary services such as composition analysis and dependency analysis. In many cases a participant acts as both a supplier and consumer, operating somewhere in the middle of a supply chain.

Participants follow a set of network rules so that SBOM systems function at scale. Suppliers create SBOMs for components the suppliers develop themselves, and suppliers define these components. For upstream components, suppliers obtain SBOMs from the appropriate upstream suppliers. If upstream SBOMs are not available, the supplier or other authors can create SBOMs, even when this involves making up or omitting baseline attributes.

An SBOM must list at least one primary component, which defines the subject of the SBOM. An SBOM lists components that can be:

1. Originally created 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

As part of delivering components to users, suppliers also deliver the associated SBOM(s), or provide 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.

A set of many interconnected supply chains is likely a directed acyclic graph, as shown in Figures 1, 2, and 3. Ultimate upstream suppliers only create original components and do not include components (i.e., do not have dependencies) from any other supplier. In Section 2.6, Carol is an example of such a supplier. Components flow downstream along supply chains throughout the graph. At the far ends of the graph, ultimate consumers only obtain components and SBOMs and do not produce components or SBOMs. Throughout the middle of the graph, most participants act as both suppliers and consumers. Even end-user organizations may act as

[26] https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-sbom-access-02
[27] https://datatracker.ietf.org/doc/html/rfc8322
[28] https://www.openchainproject.org
[29] https://datatracker.ietf.org/doc/html/rfc8520
[30] https://tools.ietf.org/html/rfc4287

suppliers, producing SBOMs for in-house components or external components such as websites, mobile applications, or IoT devices.

Suppliers are responsible for components they create and 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 comparatively low costs to generate and share that information.[31] This model distributes the cost to produce SBOM information to suppliers.

In this network, there are some scenarios where a supplier may create SBOMs for upstream components, in which the supplier is acting as the author of the SBOM and not the supplier of the upstream component. When a supplier creates such SBOMs, the supplier is expected to clearly convey that they are only the author of the SBOM, and are not the supplier of the component. This informs consumers of the lack of first-hand, authoritative SBOM information for the component. In such a case, the Author Name and Supplier Name would be different.

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

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 a user of two software products (primary components) from two different supply chains. The user has two SBOMs, one shown in Table 4 and one in Table 6.

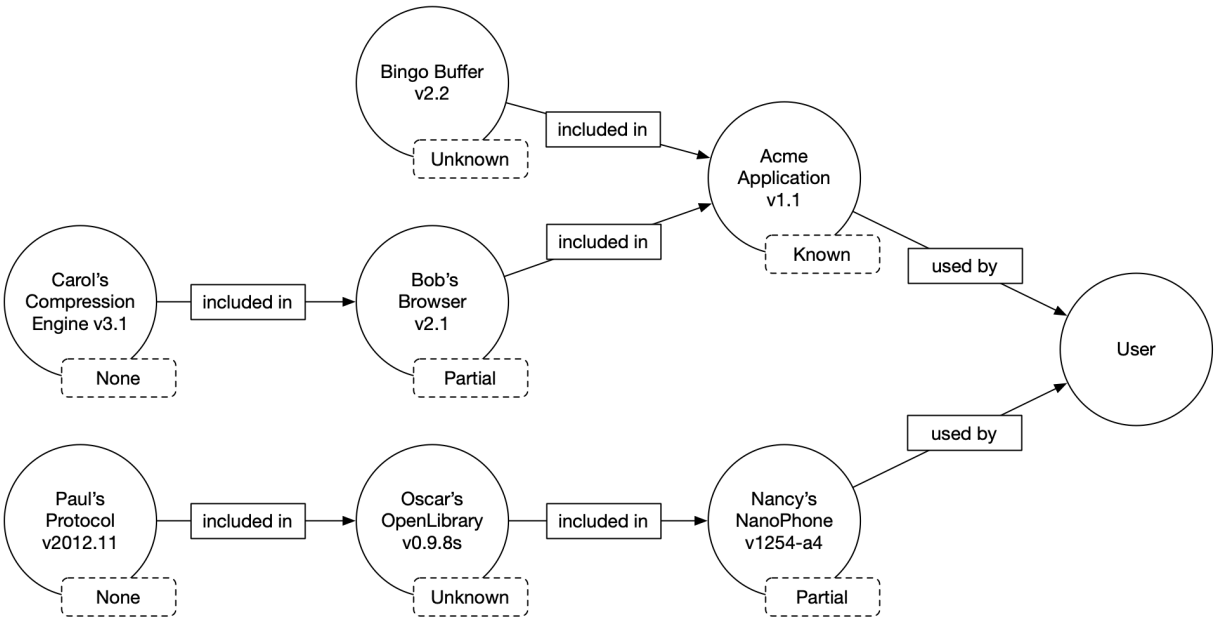

Figure 3: User graph with two supply chains

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Assertion
NanoPhone	Nancy	v1254-a4	Nancy	0x523	237	Primary	Partial
|--- OpenLibrary	Oscar	0.9.8s	Nancy	0xA23	394	Included in	Partial
|--- Protocol	Paul	2012.11	Nancy	0xB53	934	Included in	None

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

3.5 Roles and Perspectives

3.5.1 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.[32]

3.5.1.1 Produce

This SBOM model is designed with the idea that all suppliers create SBOMs for their own components. When SBOMs for upstream components are not available, a supplier may need to author an SBOM for a different supplier's component. In the case where a supplier does not provide SBOM information, 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.

3.5.1.2 Choose

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

3.5.1.3 Operate

In any industry, operators struggle with the 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 component'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.

3.6 SBOM Use Cases

The core focus of this SBOM model and the baseline attributes is to identify components and their relationships. Most SBOM use cases require additional information such as new attributes,

[32] https://ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf

relationship types, or external data connections. This section highlights several notable SBOM use cases.

3.6.1 Vulnerability Management and VEX

Vulnerability management is one of the more prominent SBOM use cases. Today, it is often an expensive and time-consuming effort to determine whether a vulnerable upstream component is used, and if the vulnerability is present or exploitable in downstream components. SBOM and VEX data helps suppliers, users, and other defenders more quickly and accurately assess the risk posed by vulnerable components, which are often hidden behind opaque supply chain relationships.

Vulnerability management requires sources of vulnerability information (such as CVE and the NVD), mapping of vulnerabilities to components (such as CPE as used in the NVD), and a way to convey vulnerability or exploitability status (such as VEX). While VEX was developed to address the vulnerability management use case, VEX is not limited to use with SBOMs or necessarily expected to be included in the SBOM itself. One concern is the incorrect detection of vulnerabilities based on limited information such as version strings, protocol banners, or other heuristics. VEX can be used to indicate that software is not vulnerable or exploitable, even when the SBOM indicates the presence of vulnerabilities in upstream components. This can save suppliers and users the costs of managing, producing, and applying security updates for components that are not affected.

3.6.2 Intellectual Property

There are a number of intellectual property use cases 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 two common use cases. A notable market exists for software composition analysis tools to help determine the contents of components, in part to identify license requirements. SBOM data would improve knowledge about composition without depending on binary analysis tools. Both SPDX and SWID were initially designed to convey license and entitlement information.

This use case requires 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.

3.6.3 High Assurance

High assurance of the source and integrity of components requires information about the pedigree and provenance of components, such as how they were built and packaged, who created and modified them, and their chain of custody through the supply chain. As with the

other use cases, high assurance will require additional attributes about components, different relationship types, and likely different supplier information.

3.7 Tool Support

The availability of SBOM generation and management tools will be critical for widespread adoption. Available tools include swid-tools,[33] swidGenerator,[34] CycloneDX Tool Center,[35] and SPDX Tools.[36] SBOM functionality will need to be integrated into software development, packaging, and asset management systems.

[33] https://apps.fedoraproject.org/packages/swid-tools
[34] https://github.com/strongswan/swidGenerator
[35] https://cyclonedx.org/tool-center/
[36] https://spdx.dev/resources/tools/, https://tools.spdx.org/app/

4 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.

4.1 SBOM

(Software Bill of Materials)

list of one or more identified components, their relationships, 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.

4.2 Component

unit of software defined by a supplier or author

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. While “component“ is primarily intended to represent packaged, compiled, or binary code, source code is not excluded, nor is hardware.

Depending on perspective in the supply chain, a component (often the primary component) can be considered to be a product, intermediate good, final good, or final assembled good.

4.3 Attribute

characteristic of or information about a component or an SBOM

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.

4.4 SBOM Entry

component and its associated attributes

In a table, an entry is a row.

4.5 Author

entity that creates an SBOM

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).

4.6 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 of software components are also consumers of other suppliers' software components. Suppliers and consumers can exist within the same organization.

4.7 Consumer

entity that obtains or receives SBOMs

An entity can be, and often is, both a supplier and consumer, existing in the middle of a supply chain.

5 Background

5.1 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.[37]

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 baseline SBOM attributes, (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.

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

5.2 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.[38]

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.

5.3 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 but not limited to automotive, financial, healthcare, operational technology (OT), and “traditional” IT. Despite the focus on software (the “S” in SBOM), 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 outside of this scope in order to more fully realize benefits of software component transparency:

- Lists of known vulnerabilities mapped to components
- Ways to convey the status or degree of exploitability or exposure of vulnerabilities (VEX)
- Standardized sharing mechanism for SBOMs[39]

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.

[38] https://www.ntia.doc.gov/SoftwareTransparency
[39] https://ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf

6 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 software transparency, enable businesses to better manage the security of their networks, and allow suppliers to monitor their components 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 has been defined. These attributes generally align with existing formats such as SPDX, CycloneDX, 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 the use of SBOM matures and becomes more common, the ready availability of baseline SBOM information will 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.

7 Changes

Significant changes between this and the previous edition include:

- Added Timestamp to Baseline Attributes
- Clarified requirements aspects of Baseline Attributes
- Added CycloneDX as an additional format
- Removed Existing Formats (previously Section 3), renumbered accordingly
- Updated language in Baseline Attributes and Terminology
- Updated and harmonized language across working groups
- Updated figures and tables
- Made various editorial improvements and clarifications

8 Acknowledgements

The chairs of the Framing Working group, Michelle Jump (MedSec) 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.

Tom Alrich, Tom Alrich LLC
John-Luc Bakker, BlackBerry US
Stephen Barrett, BlackBerry US
David Dillard, Veritas Technologies
Dick Brooks, Reliable Energy Analytics LLC
Phil Englert, MedSec
Christopher Gates, Velentium
Les Gray, Abbott
Charlie Hart, Hitachi
Audra Hatch
Ed Heierman, Abbott
JC Hertz
Eliot Lear
Bruce Lowenthal, Oracle
Bob Martin, MITRE Corporation
Dmitry Raidman, Cybeats
Vijay Sarvepalli, CERT Coordination Center
Duncan Sparrell, sFractal Consulting
Rich Steenwyk, GE Healthcare
Kate Stewart, The Linux Foundation
Tim Walsh

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