Source: Pierre Metivier (https://flic.kr/p/rdHr) CC BY-NC / Desaturated from the original

CISA Hosted
SBOM-a-rama
June 14, 2023

SBOM Tooling & Implementation Work Stream

Tooling and Implementation Work Stream

Co-chairs: Melissa Rhodes & Kate Stewart

Meeting since August 25, 2022

- Thursday at 1500 EDT
- Contact SBOM@cisa.dhs.gov to be added to mailing list and meeting invite

Discussion of SBOM tooling implementation pain points and propose strategies to improve interoperability

Melissa Rhodes (Virtual)

Medtronic

SBOM Program Manager

melissa.m.rhodes@medtronic.com

Kate Stewart (In Person)

Linux Foundation

VP of Dependable Embedded Systems

stewart@linux.com

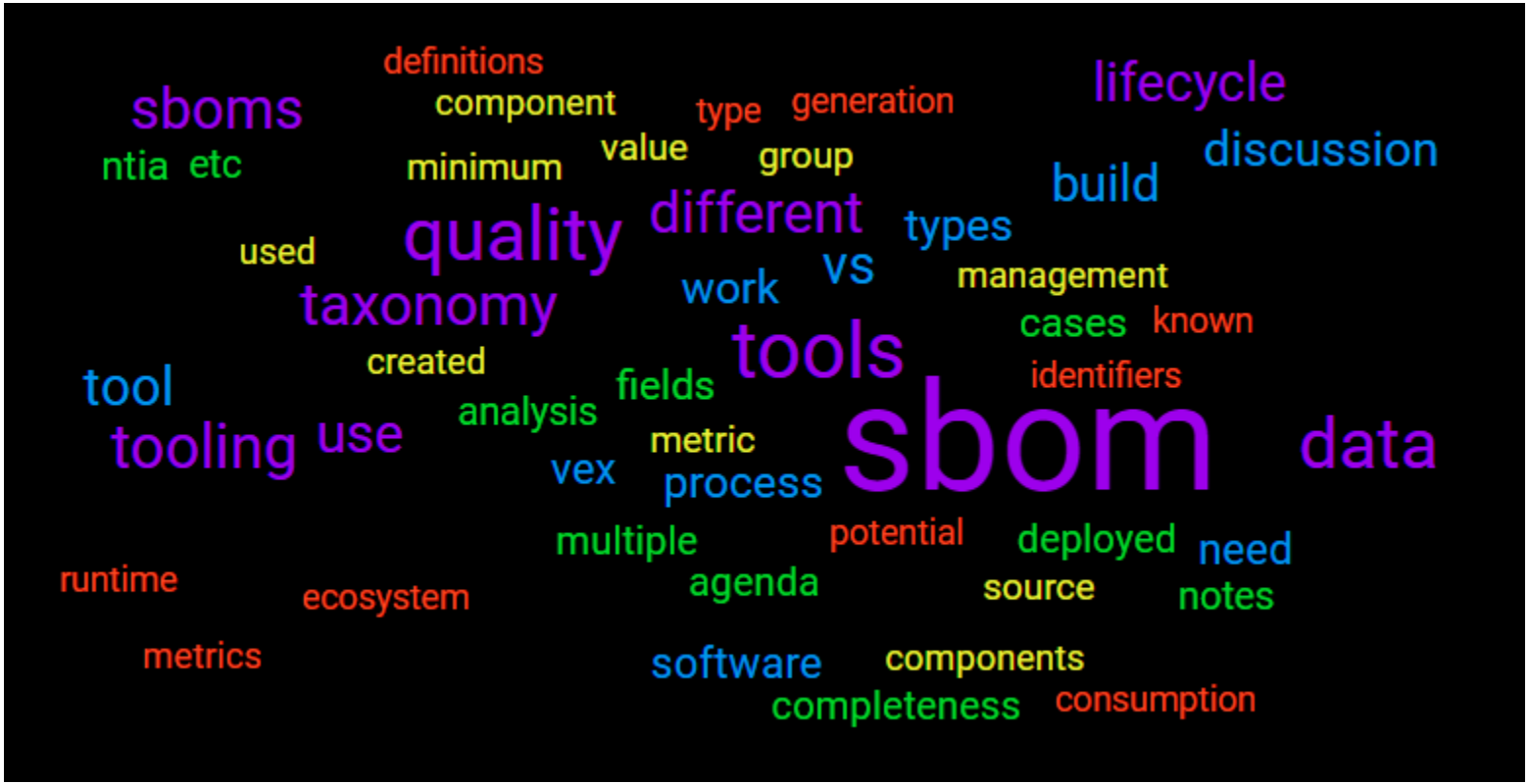
Outline

- History (last SBOM-a-rama to now)
  - Group understanding of current pain points for deploying at scale
  - Transition SBOM-types discussion from openSSF
- Publications
  - SBOM types
- Results of survey topics
- SBOM Field Definitions - building up pragmatic guidance
  - Format adherence (syntax)
  - What is filled → (semantics)
  - “Red areas” for where problems”
  - Pragmatic Practices - for field contents.
- Next Steps
  - Publish of pragmatic practices for minimum fields
  - Plugfests?

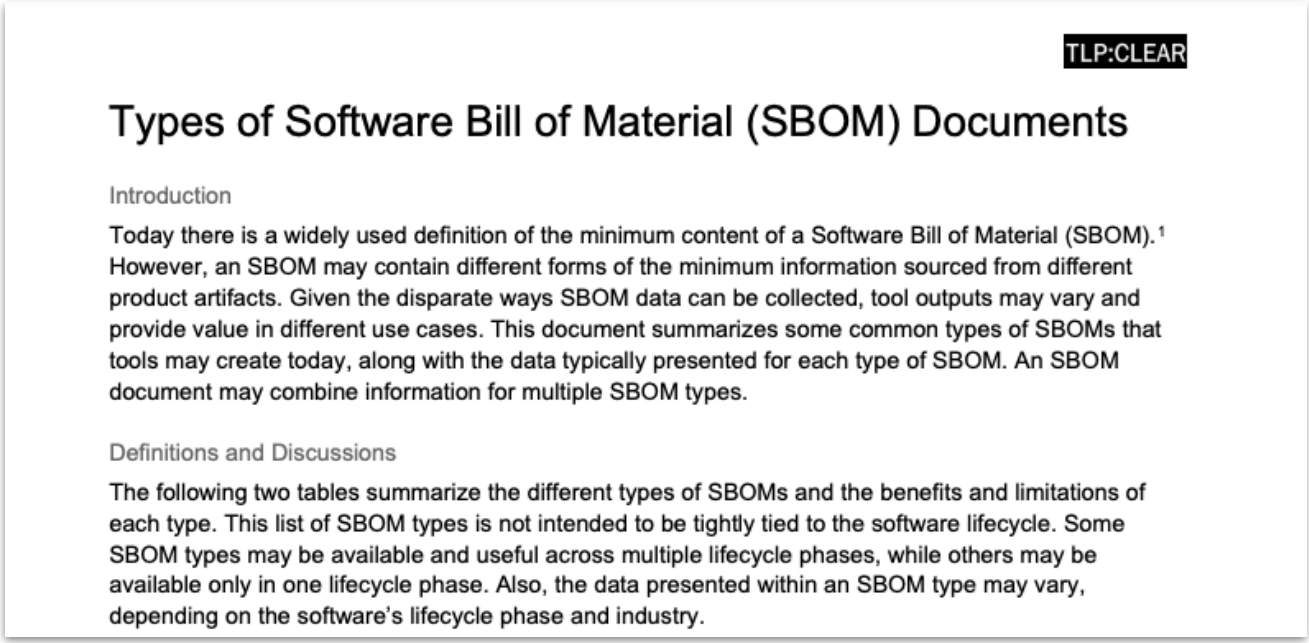
2022/8/25 → SBOM-a-rama

- Group reforming & brainstorming
- Discussion of pain points in industry
- Collaboration with OpenSSF “SBOM everywhere”
  - refine distinction between how different tooling produces different SBOM contents → types
- Working on SBOM Types
  - Definitions, data, pros & cons
- Brainstorming and Survey for next topic
- Pragmatic practices for filling in SBOM minimum elements

Brainstorming: Challenges for SBOM Adoption



Publication



TLP:CLEAR

Types of Software Bill of Material (SBOM) Documents

Introduction

Today there is a widely used definition of the minimum content of a Software Bill of Material (SBOM).[1] However, an SBOM may contain different forms of the minimum information sourced from different product artifacts. Given the disparate ways SBOM data can be collected, tool outputs may vary and provide value in different use cases. This document summarizes some common types of SBOMs that tools may create today, along with the data typically presented for each type of SBOM. An SBOM document may combine information for multiple SBOM types.

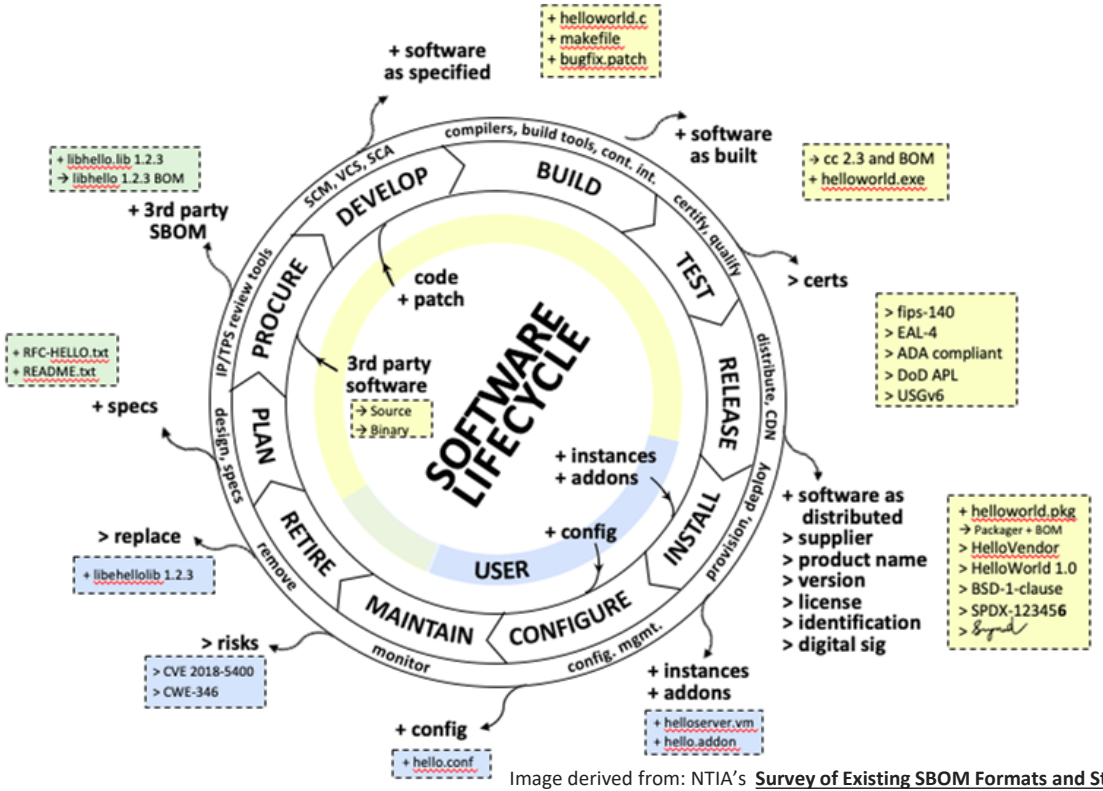
Definitions and Discussions

The following two tables summarize the different types of SBOMs and the benefits and limitations of each type. This list of SBOM types is not intended to be tightly tied to the software lifecycle. Some SBOM types may be available and useful across multiple lifecycle phases, while others may be available only in one lifecycle phase. Also, the data presented within an SBOM type may vary, depending on the software's lifecycle phase and industry.

Source: https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

When should an SBOM be created or consumed?

SBOMs provide a mechanism to track **key artifacts** and **dependencies** as well as other useful **configuration management information** throughout the software lifecycle.



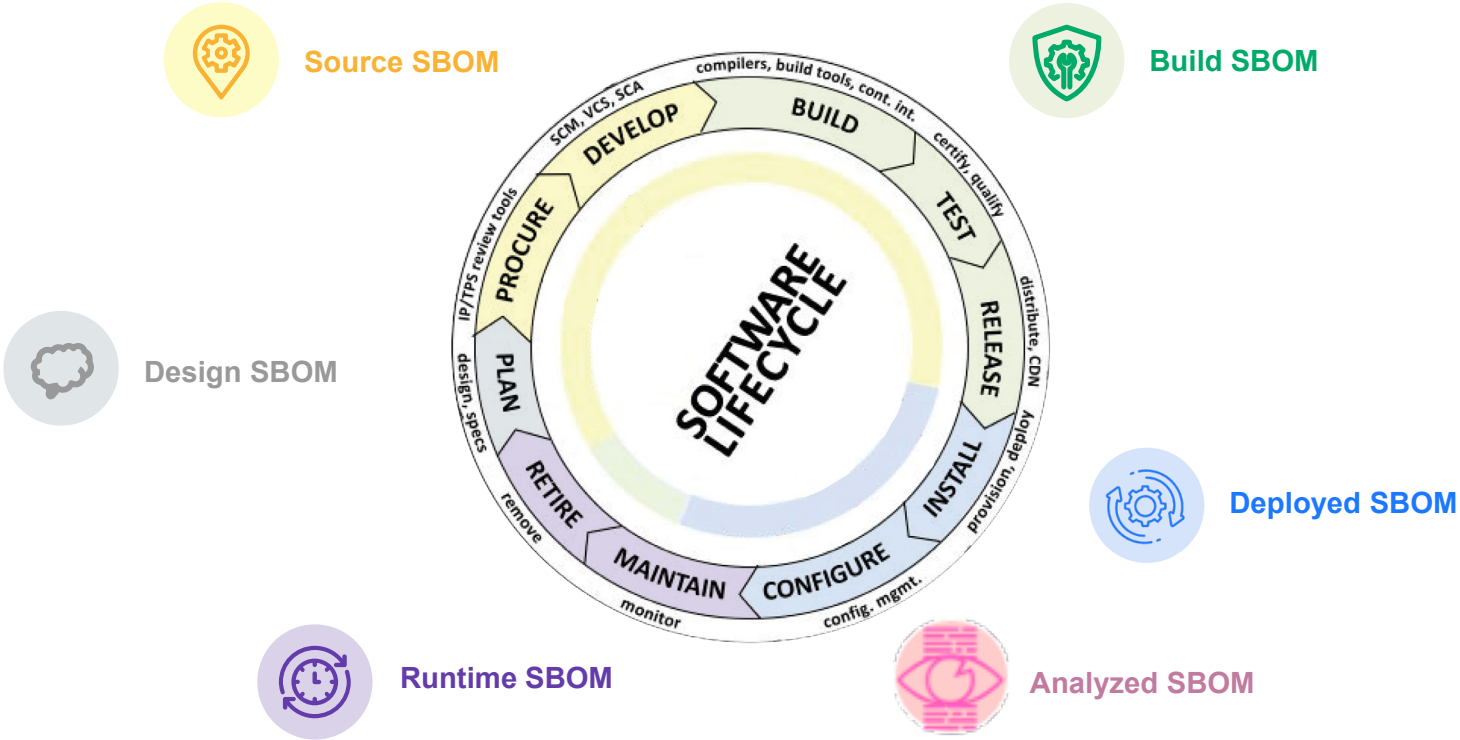
LEGEND
+ material
> metadata
→ reference
example BOM fragment
supplier
consumer
other

Image derived from: NTIA's **Survey of Existing SBOM Formats and Standards**

SBOM TYPE	DEFINITION
Design	SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.
Source	SBOM created directly from the development environment, source files, and included dependencies used to build an product artifact.
Build	SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.
Deployed	SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.
Runtime	BOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an “Instrumented” or “Dynamic” SBOM.
Analyzed	SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a “3rd party” SBOM.

Source: Types of Software Bills of Materials (SBOM) published by CISA on 2023/4/21

KEY: The data to generate SBOMs accurately shows up in different phases of development and operations.



Participant Survey: Topics Interest & Poll Results

1. **SBOM quality** [Interest level:20]
    - Voted Top priority: **14**
2. **SBOM inventory detection types** [Interest level: 11]
    - Voted Top Priority: **11**
3. **Tooling that provides protection and accessibility of SBOM data** [Interest level: 9]
    - Voted Top Priority: **9**
4. **Enterprise deployment of SBOM tools at scale** (creation, collection, and dissemination) [Interest level: 12]
    - Voted Top Priority: **6**
5. **Data management tooling** [Interest level: 11]
    - Voted Top Priority: **4**
6. **Versioning**: Versions of implementation formats [Interest level: 8]
    - Voted Top Priority: **3**

SBOM Quality - Work In Progress

- Structural correctness
  - Syntactic correct for the format used
  - Compliance with specification
- Content is accurate
  - Completeness of at least the minimum SBOM elements
  - Compliance to description of field (**challenge is different understanding of what a field means**)
  - Lots of way to be syntactically accurate, but convey no useful information - key fields with NOASSERTIONS
- Context to the data
  - Identifiers from 1st and 3rd party perspectives, names, hashes, etc.
  - Dependencies - what is “good enough”?
- Process and Timeliness

Quality Category	Detail	Notes
Structural	SBOM is presented in community-supported specific versions	SPDX, CycloneDX, and SWID specific versions
Structural	SBOM adheres to the underlying specification	The SBOM structure matches the underlying schema
Semantic	SBOM includes all fields required by the underlying specification	The specification-based checks
Quality	SBOM includes optional fields that help with the identification of data or improve automation: - Package Version - Package Checksum - Vulnerability lookup identifiers - Primary Package Purpose - Package Copyright Text - License Identifier/details for non-SPDX license	An SBOM with “more” data (even the optional fields) reduces the need for regeneration if the use case changes
Validation & Completeness	Additional non-SBOM data that helps validate the SBOM’s specific workflow - Verifiable link to the underlying manifest, product and its provenance - Digital signatures of the SBOM - Signature verification mechanism described in/outside of the SBOM - Transitive dependencies included (or declared to be unknown) - Vulnerability assessment (including date/time and vulnerability data sources)	Supplemental data that can enable verifying the provenance of SBOM itself, its connection to underlying product, assertion of its components and any previous analysis done for the state of vulnerabilities
Open Quality Considerations	1. Quality drift based on tool generator’s fixes - any way to capture that within SBOM/Outside of SBOM? 2. Packages modified from their public presence and used in product without FilesAnalyzed and therefore have potential to be problematic despite familar package identifier	

Work in Progress Resources: [CISA SBOM Quality - Working Notes]

What is “Quality data” for SBOM Field Descriptions

Data Field	Description
Supplier Name	The name of an entity that creates, defines, and identifies components.
Component Name	Designation assigned to a unit of software defined by the original supplier.
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version.
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases.
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y.
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.
Timestamp	Record of the date and time of the SBOM data assembly.

Source:https://www.ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf.
See also US Executive Order on Cybersecurity section 4(f)

WIP: Common Understanding of SBOM Elements

Supplier Name	The name of an entity that creates, defines, and identifies components.
Component Name	Designation assigned to a unit of software defined by the original supplier.
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases.
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y
Author of SBOM Data	The name of the entity that creates the SBOM data for this component
Timestamp	Record of the date and time of the SBOM data assembly
Root of Dependencies	A piece of software can be represented as a hierarchical tree, made up of components that can, in turn, have subcomponents, and so on.
Level of Dependencies	At a minimum, all top-level dependencies must be listed with enough detail to seek out the transitive dependencies recursively.
Known Unknowns	For instances in which the full dependency graph is not enumerated in the SBOM, the SBOM author must explicitly identify “known unknowns.”
Hash of the Component.	A cryptographic hash would provide a foundational element to assist in this mapping, as well as helping in instances of renaming and whitelisting.

Example: WIP: Pragmatic Practice for Supplier Name

Was the **component modified** while in your **possession**?

Yes - list the component with **your organization** in the supplier name field and consider listing who supplied the component as dependent relationship in the SBOM

No - list the component with **your supplier's name** as indicated for commercial or open-source entities

Commercial source: list the commercial supplier's "legal entity" name

Open-source: at a minimum, list project name.

- Consider listing the host foundation before the project name (Ex: Apache Tomcat)

What does it mean for a component to be **modified**?

When one or more of the following are true:

- Supplied source code or logic has been altered
- Component hash is different than the one supplied
- Supplied code is compiled into another component
- Tool settings have been changed
- Configurations or optimization options are invoked

Next Steps

Finish Pragmatic advice document for filling in fields and publishing SBOMs.

Review if pain points highlighted from last poll is still accurate.

- If so, start working on next priority
- If not, determine topics to add, and redo poll to determine point to focus.

Plugfests?

Questions?

Contact SBOM@cisa.dhs.gov to be added to mailing list and meeting invite if you want to join in the discussion.