Federal Office
for Information Security

Technical Guideline TR-03183: Cyber Resilience Requirements for Manufacturers and Products

Part 2: Software Bill of Materials (SBOM)

Document history

Table 1: Document History

Version	***Date***	***Description***
1.0	2023-07-12	Version of TR-03183-2 for first publication
1.1	2023-11-28	English translation; updates in the requirements for creator, version and licence information

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn
E-Mail: TR03183@bsi.bund.de
Internet: https://bsi.bund.de/dok/TR-03183

Table of Contents

1 Introduction

This Technical Guideline describes the requirements for a “Software Bill of Materials (SBOM)”. An SBOM is a machine-processable document and corresponds to an electronic bill of materials / parts list. It inventories a code base and thus contains information on all components used in a software. This information can be presented in different breadth and depths - ranging from a basic structure to a fine-granular breakdown of products and software components. Different formats exist for the representation and distribution of an SBOM.

An SBOM should be used by every software creator and provider in order to be able to transparently represent software complexity and to know which components (e. g. libraries) are used, since a variety of different sources and components are usually used. This knowledge is essential for software management processes, especially for a continuous IT security process and the lifecycle management of software; it is therefore considered “best practice” for a secure software supply chain. An SBOM can be public or non-public and can be distributed in different ways. Typically, software creators use one or more third-party components. They create and manage the SBOMs of their own software; likewise, they take the consumer role of the SBOMs of the included components. The abundance of SBOM information and the possible differences in the structure of SBOMs mean a great deal of work for each creator. Only automation can effectively address that.

SBOM information can be used to check whether a product is potentially affected by a vulnerability by comparing its component list with the software components listed in a vulnerability database. However, an SBOM does not contain any statement regarding vulnerabilities or their exploitability. SBOM data is static with respect to software that is not changing, while vulnerability information will change. Whether and to which extent a vulnerability of an integrated software component poses a risk to the product described by the SBOM is also not provided. This requires further information on the specific vulnerability, for example by means of security advisories or VEX[1] (Vulnerability Exploitability Exchange).

In order to confirm whether a product is affected by a vulnerability or not, it is necessary to compare the SBOM of the product with vulnerability information, such as the CVE (Common Vulnerabilities and Exposures) or security advisories of the component manufacturers or providers. Furthermore, an analysis of the software itself is necessary to determine how its potentially affected subcomponents are used, and thus whether and how one's own software is affected. This must be carried out as part of the vulnerability management for the product. The result of this analysis is then provided to the users of the software as a security advisory or VEX for the product.

Originally, SBOMs were mainly used for license management. This Technical Guideline also specifies this use case in an interoperable manner, in order to provide a complete set of requirements for the common use cases of SBOMs.

In the current draft of the Cyber Resilience Act (CRA)[2] the compilation of an SBOM is mandatory. The CRA is a market access regulation of the EU for products with digital elements, which obliges their providers to continuously operate a vulnerability management process and to provide information on their products in a transparent and comprehensible form. In the US, an SBOM is already required for software acquired by the Federal government by the US Executive Order 14028 of May 2021[3] and the FDA (Food and Drug Administration) demands an SBOM to be submitted as part of the approval of new medical devices since March 2023[4].

[1] https://www.ntia.gov/files/ntia/publications/vex_one-page_summary.pdf
[2] https://ec.europa.eu/transparency/documents-register/detail?ref=COM(2022)454&lang=en
[3] https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/ Section 4
[4] https://www.congress.gov/117/bills/hr2617/BILLS-117hr2617enr.pdf Section 3305

2 Formalia

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

3 Basics

3.1 Definition SBOM

A “Software Bill of Materials” (SBOM) is a machine-processable file containing supply chain relationships and details of the components used in a software product. It supports automated processing of information on software components. This covers both the so-called “primary component” and used (third-party) components.

An SBOM MUST contain certain minimum information (see chapter 5), but MAY be expanded and detailed as desired.

A new, separate SBOM MUST be generated for each software version. A new version of the SBOM for a given software version MUST be generated if and only if more information on the included software components is provided or errors in the SBOM data are corrected.

An SBOM does not contain vulnerability information, because SBOM data is static with respect to software that is not changing. Nonetheless, the formats for describing an SBOM may offer the ability to include vulnerability information.

3.2 Terms used

A “software component” for the purposes of this Technical Guideline MUST correspond to a single executable file[5].

In this Technical Guideline, a distinction is made between “Vendor” (in German: “Anbieter”; alternatively, but not necessarily with a commercial background: “Supplier”, in German: “Lieferant”) and “Creators” (in German: “Ersteller”, alternatively, “Authors” / “Autoren”), since “Manufacturer” may be interpreted as combining these two roles in the sense of “vendor of self-created software” (as the German term “Hersteller” is usually interpreted).

[5] see also https://en.wikipedia.org/wiki/Executable

4 SBOM formats

An SBOM MUST be in a format that meets one of the following specifications in one of the specified versions.

- CycloneDX[6], version 1.4 or higher
- Software Package Data eXchange (SPDX)[7], version 2.3 or higher

[6] https://cyclonedx.org/specification/overview/
[7] https://spdx.dev/specifications/

5 Content requirements

5.1 Level of detail

For an SBOM that is compliant with this Technical Guideline, recursive dependency resolution MUST be performed at least for each component included in the scope of delivery on each path at least up to and including the first component that is out of the scope of delivery (see also chapter 6.2.4). This SBOM MUST be created as part of the build process or its equivalent where the build process does not exist (Build SBOM, see also chapter 6.3.3).

5.2 Required data fields

5.2.1 Required data fields for the SBOM itself

Each SBOM MUST contain at least the following information:

Table 2: Required data fields for the SBOM itself

Data field	***Description***
Creator of the SBOM	Email address of the entity that created the SBOM. If no email address is available this MUST be a "Uniform Resource Locator (URL)".
Timestamp	Date and time of the SBOM data compilation according to the specification of the formats (see chapter 4)

5.2.2 Required data fields for each component

For each component included in an SBOM, at least the following information MUST be provided[8]:

Table 3: Required data fields for each component

Data field	***Description***
Component creator	Email address of the entity that created and, if applicable, maintains the respective software component. If no email address is available this MUST be a "Uniform Resource Locator (URL)".
Component name	Name assigned to the software component by its creator
Component version	Identifier used by the creator to specify changes in the software component to a previously created version. Identifiers according to Semantic Versioning[9] or alternatively Calendar Versioning[10] SHOULD be used if one determines the versioning scheme. Existing identifiers MUST NOT be changed for this purpose.
Dependencies on other components	Enumeration of all components on which this component is directly dependent, according to the requirements in section 5.1

[8] cf. https://www.ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf sections IV and V as well as https://www.ntia.gov/sites/default/files/publications/ntia_sbom_framing_2nd_edition_20211021_0.pdf section 2.2 and https://www.ntia.gov/files/ntia/publications/ntia_sbom_formats_energy_brief_2021.pdf page 5.

[9] https://semver.org/

[10] https://calver.org/

Data field	*Description*
Licence	Associated licence(s) of the component from the perspective of the SBOM creator. Principles of licence identification: • Licences MUST be identified with their SPDX identifier. The indication is made with the so-called identifiers for licences and expressions[11]. • Indication of non-standardised licence identifiers: If licence identifiers cannot be found in the list from SPDX, the licence database Scancode LicenseDB AboutCode[12] MUST be consulted next. Identifiers from this database use the prefix `LicenseRef-scancode-[...]` in their SPDX identifier to map the origin of the specified identifier. • Completely unknown licences: If a licence discovery is not available in any of the established identifier lists, the prefix `LicenseRef-<licence_inventorising_entity>-[...]` MUST be used according to "Annex D: SPDX License Expressions"[13] to assign a unique (per LicenseRef namespace) licence identifier. Principles of licence similarity: • Licences are similar if they can be mapped to another licence according to "Annex B: License Matching Guidelines and Templates"[14]. • Placeholders and templates in licence texts: If placeholders and templates exist in licence texts, a replacement of these placeholders and templates by individualised content MUST NOT be understood as a licence modification, but MUST be assigned to the same licence identifier. • SPDX operators and licence concatenations: Licence expressions of multiple licensing, licence choices and licence exceptions MUST be mapped with SPDX operators. The licence operators link licence identifiers. Permitted operators MUST be selected according to "Annex D: SPDX License Expressions". • Exception clauses for licences: If an exception clause is attached to a licence text, it MUST be attached to a licence identifier with `WITH` according to the allowed SPDX operators. The names of the exception clause MUST be described with identifiers analogous to the requirements on licence identification[15]. • Text modifications: If a wording in a licence text is slightly modified compared to a licence text of a licence with a known licence identifier, this licence identifier SHOULD be used for the modified licence if the modification is not substantial. Examples are the addition or removal of liability-clauses or remarks to existing trademarks.

[11] https://spdx.org/licenses/
[12] https://scancode-licensedb.aboutcode.org/
[13] https://spdx.github.io/spdx-spec/v2.3/SPDX-license-expressions/
[14] https://spdx.github.io/spdx-spec/v2.3/license-matching-guidelines-and-templates/
[15] https://spdx.github.io/spdx-spec/v2.3/SPDX-license-list/#a2-exceptions-list

Data field	*Description*
Hash value of the executable component	Cryptographically secure checksum (hash value) of the component in its executable form (i. e. as the single executable file on a mass storage device) as SHA-256

5.3 Additional data fields

5.3.1 Additional data fields for the SBOM itself

Each SBOM MUST additionally include the following information, if it exists and its prerequisites are fulfilled:

Table 4: Additional data fields for the SBOM itself

Data field	*Description*
SBOM-URI	"Uniform Resource Identifier (URI)" of this SBOM

5.3.2 Additional data fields for each component

For each component included in an SBOM, the following information MUST additionally be provided, if it exists and its prerequisites are fulfilled:

Table 5: Additional data fields for each component

Data field	*Description*
Source code URI	"Uniform Resource Identifier (URI)" of the source code of the component, e. g. the URL of the source code repository
URI of the executable form of the component	"Uniform Resource Identifier (URI)", which points directly to the executable form of the component.
Hash value of the source code of the component	Cryptographically secure checksum (hash value) of the component source code[16] as SHA-256
Other unique identifiers	Other identifiers that can be used to identify the component or to look it up in relevant databases, such as Common Platform Enumeration (CPE) or Package URL (purl).

[16] The method to calculate the hash value of the source code is currently not specified.

6 Annex

6.1 Explanations

This section provides more information on certain terms or definitions.

6.1.1 Machine-processable

SBOMs are defined as machine-processable files in chapter 3.1. This implies that machines can create, read, modify, process, analyse and evaluate the content and act based on the data. The content itself is well-defined and structured. The term “machine-readable” can be interpreted in multiple ways and is therefore not used.

6.1.2 Components

In chapter 3.2 a “software component” corresponds to a single executable file. Executable file means code that is executed by a computer, either directly or by a runtime system. Examples include compiled binaries (“executables”), scripts (e. g. Python, Shell) and shared libraries. Not covered by the term are, e. g. configuration files, graphic files, documentation.

6.1.3 Scope of delivery

The scope of delivery of a software product comprises all parts of software, originated by the supplier or a third party, that are delivered with the software product. Not included by this term are parts of software that have to be acquired or obtained separately.

6.1.4 Licence Information

The required fields for every component include licence information. On the one hand handling licence information is a common use case for SBOMs. On the other hand licence information can be crucial in handling vulnerabilities, e. g. if the licence of a component does not allow the user to modify the code to mitigate a vulnerability.

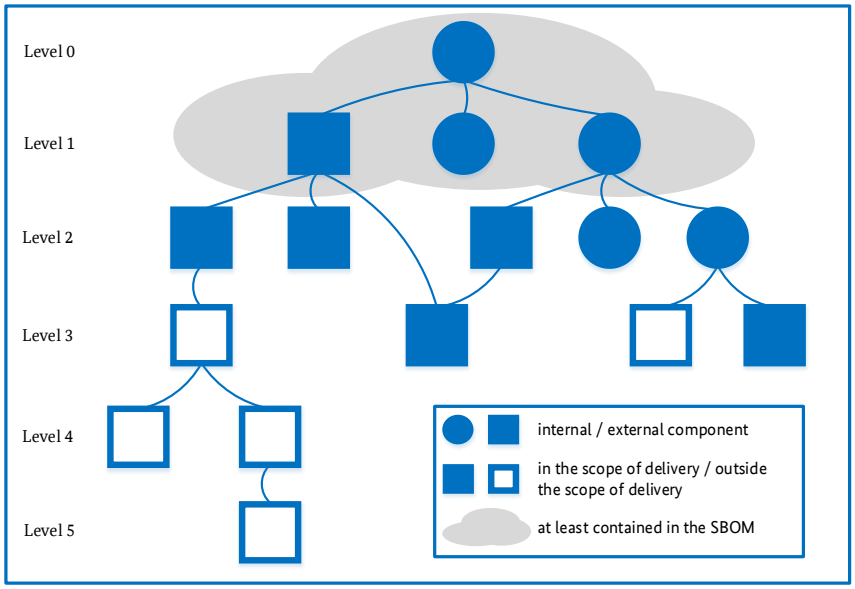
6.1.5 Vulnerability Information

The SBOM definition in chapter 3.1 states that vulnerability information is not contained in an SBOM. Information on vulnerabilities of a certain version of a software changes over time while the crucial information of an SBOM (e. g. on dependencies) is static. If vulnerability information is included in an SBOM, this static data is unnecessarily propagated along the software supply chain in unaltered form each time the vulnerability information is updated. Consequently, it is strongly recommended not to include vulnerability information in an SBOM, even though an SBOM format specification supports that. The recommended format for distributing vulnerability information is CSAF (including also VEX as a profile).

6.2 Level of detail of an SBOM

An SBOM can be created in different depths, e. g. according to the following classification.

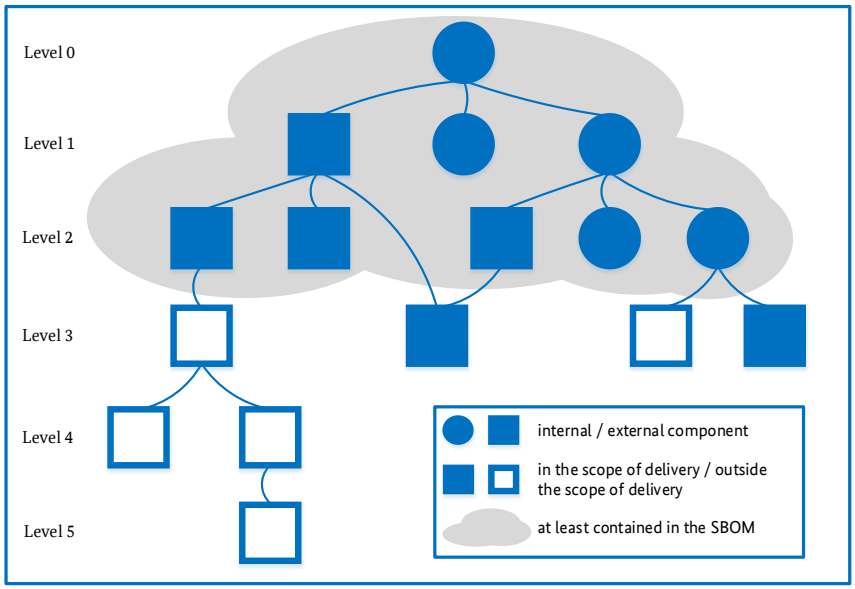
6.2.1 Top-level SBOM



In addition to the full description of the primary component, the SBOM contains the full description of all components, which the primary component directly depends on.

Figure 1: Top-level SBOM

6.2.2 *n*-level SBOM



In addition to the full description of the primary component, the SBOM contains the full description of all components, which are directly or transitively depended upon via *n* levels by the primary component. This means that the recursive resolution of the transitive dependencies is limited to *n* steps in depth. If the path from the primary component is shorter than *n* levels, all components on this path have to be resolved and, consequently, fully described.

A top-level SBOM is a 1-level SBOM.

Figure 2: n-level SBOM

6.2.3 Transitive SBOM

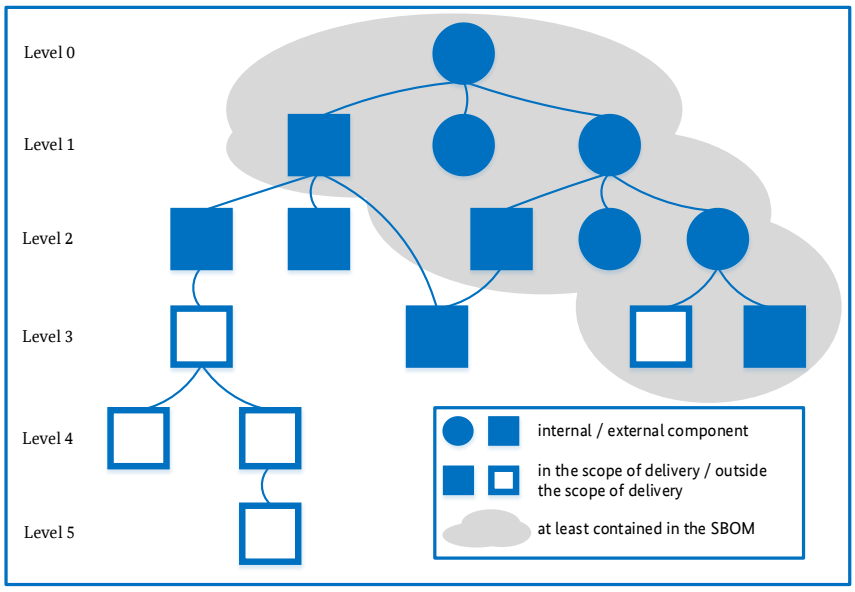


Figure 3: Transitive SBOM

In addition to the full description of the primary component, the SBOM contains information on at least all components, which are directly or transitively depended upon by the primary component. The full description and recursive resolution of components and their dependencies is performed on each path at least up to and including the first external component (i. e. third-party component). This component must also be fully described in the SBOM except for the dependencies of this component. The dependencies of external components do not have to be resolved.

Compared to an *n*-level SBOM with a first external component on a path at level *n-1* the transitive SBOM contains less information on this external component.

6.2.4 Delivery item SBOM

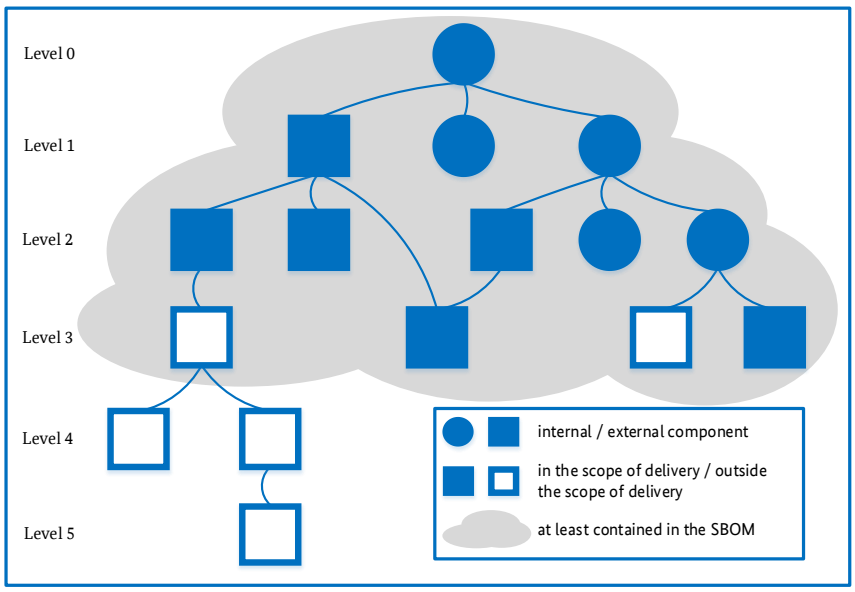
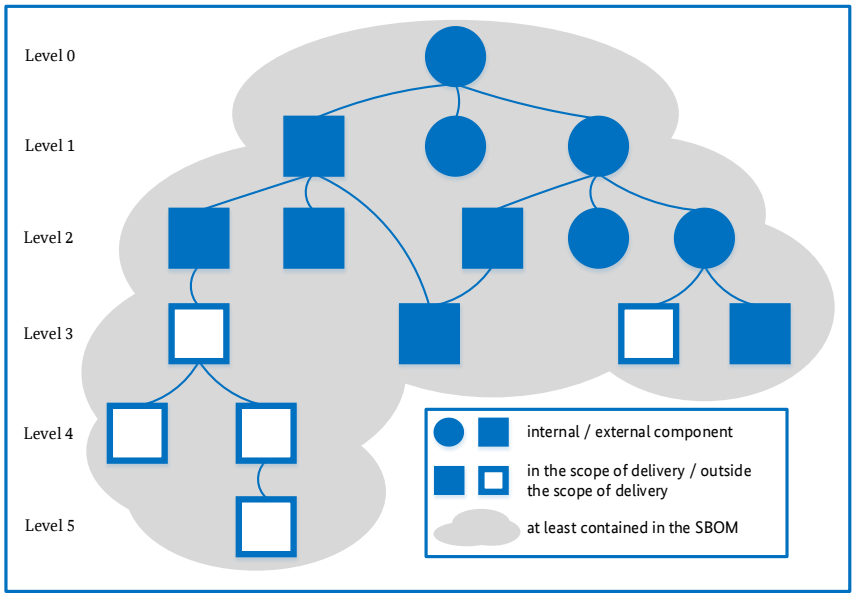


Figure 4: Delivery item SBOM

In addition to the full description of the primary component, the SBOM contains the full description of at least all components, which belong to the scope of delivery and are directly or transitively depended upon by the primary component. The full description and recursive resolution of components and their dependencies is performed on each path at least up to and including the first component, which is outside the scope of delivery. This component must also be fully described in the SBOM except for the dependencies of this component; consequently, these dependencies do not have to be resolved.

6.2.5 Complete SBOM



In addition to the full description of the primary component, the SBOM contains the full description of all components, which are directly or transitively depended upon by the primary component. The full description and recursive resolution of the components and their dependencies is carried out completely.

Figure 5: Complete SBOM

6.3 SBOM classification

Depending on how or when an SBOM is created as part of the development, delivery, installation and execution process of a software component, the data differs, which is available in that specific situation. Consequently, the information that can be compiled in the SBOM also differs. A common differentiation is to distinguish between the following SBOM classes.

6.3.1 Design SBOM

The SBOM is created based on the planned set of included components of a new software artefact. The components do not have to exist yet.

6.3.2 Source SBOM

The SBOM is created from the development environment, the source files and the dependencies it uses.

6.3.3 Build SBOM

The SBOM is created as part of the build process based on e. g. source files, dependency information, already created components, volatile build process data and other SBOMs.

Notes:

- In order to enable capturing executable, binary components that already exist (i. e. precompiled code), creating a Build SBOM focuses on the linker run for translated code, not the compiler run.
- In the case of interpreted code, only the source code exists; each executable file has to be listed as a component. The interpreter has to be specified as a dependency, as far as reasonably possible.

6.3.4 Analysed SBOM

The SBOM is created after the build process by analysing artefacts such as executables, packages, containers and virtual machine images. This type is also referred to as “3rd party SBOM”.

6.3.5 Deployed SBOM

The SBOM provides an inventory of the software on a system. This can be a compilation of other SBOMs, taking into account configuration options and examination of execution behaviour in a (possibly simulated) deployment environment.

6.3.6 Runtime SBOM

The SBOM is created using the system executing the software to capture running (i. e. executing) software components as well as their external calls and dynamically loaded components at runtime only (i. e. in memory). This type may also be referred to as “Dynamic SBOM”.

6.4 Further information

6.4.1 Information from the NTIA

The “National Telecommunications and Information Administration (NTIA)” of the “United States Department of Commerce” offers a great deal of further information on the subject of SBOM at https://ntia.gov/sbom.

6.4.2 Information from CISA

The Cybersecurity and Infrastructure Security Agency (CISA) of the United States Department of Homeland Security also offers further information on SBOM at https://cisa.gov/sbom.