A Standard BOM For Siemens

T. Graf, A. Gschrei, T. Jensen
Siemens

- 310k Employees
- 47k R&D Employees
- 50+ R&D Locations
- 120k Components Used
- 20+ Software Eco-Systems (JavaScript, Java, C#, Go, Python, Lua, Swift, …)
- +25k New Components/Year
What is an SBOM - A Software Bill of Materials?

An SBOM is an inventory of components, a list of ingredients that make up a software product. It also ...

- is a formal, machine-readable document.
- includes information about the components, especially a unique identifier.
- gives the components' hierarchical relationships.
- should be comprehensive (or explicitly state where it could not be).
- may include OSS and proprietary software.
- can be widely available or access-restricted.
- should be generated automatically.

The primary purpose of an SBOM is to uniquely and unambiguously identify components and their relationships to one another.
SBOMs Are Created with a Specific Use Case In Mind

License Compliance

Use SBOMs to ensure that all obligations from OSS and other licenses are met.

- Rich and complete information preferred
- Source code required for all components(!)
- Used internally

Security Vulnerability Monitoring

Use SBOMs to enable monitoring of security vulnerabilities as they emerge.

- Slightly different fields required, such as CPEs
- Can include build tools and test frameworks
- Source code not needed
- Used internally

Regulatory

Use SBOMs to comply with regulations like U.S. EO14028 or the E.U. Cyber Resilience Act.

- Only strictly required content to minimize attack surface
- Source code not needed
- Published

All use cases have in common that the SBOM must be accurate and complete, including all transitive dependencies.
Software Bills Of Materials Are About Interoperability

Source Ecosystems
Libraries, Open-Source Software, Transitive Dependencies

Partner Companies
Components provided by partners may come with their own SBOMs.

3rd Party Vendors
Software Products, e.g. PostgreSQL, Nginx, …

Siemens Business Units
Our own internally developed Inner Source components have SBOMs.

Container Images
Built and/or consumed, Base Images, Layers

Your Product’s SBOM

… and more
The list goes on.
Nearly every team is different
- No silver bullet
- No universal automation approach

We need to have a set of tools
- In order to simplify SBOM generation / software license automation for our developers
- Inner source or open source

We need to have simple to use libraries
- When there are no specific tools
- When teams need customization
- Inner source or open source
For Us It Is Important To Get Our SBOMs Right

**Security**

Identification of vulnerable products only possible if SBOMs are accurate

Quick reaction times for 0-days are essential

Cannot mitigate if you don't even know your product is vulnerable

**Compliance**

Failure to comply with the license terms of third-party components can lead to litigation

Lawsuits are time-consuming and expensive

Claims for compensation can easily reach millions of $$$

Worst case: an injunction may prohibit the sale of affected products

Both scenarios are a PR nightmare if a company gets them wrong!
Container Images Present a Special Challenge When Creating SBOMs

**Known Base Image(s)**
The base image came with an accurate SBOM, and we add stuff to it.

**From Internet, Trustworthy**
False sense of security, because we trust the publisher of the ready-made image.

**Self-Built**
The entire image was built by our own people, all content is explicitly known.

**Complex Containers**
SBOM result of "puzzle", e.g. Debian base, ElasticSearch, custom backend module, angular, …

**Unrestricted, From Internet**
“Look, I found this nice image on Docker Hub!”

"easy"  High Effort, High Certainty  **SBOM Creation**  Low Effort, Low Certainty  "hard"
A common SBOM format and tooling for Siemens would be nice!
Standard BOM

A standardized description of a Software Bill of Materials (SBOM), plus some tooling for generation and consumption of standard-born SBOMs.

Standard BOM is:

- a subset of CycloneDX
- programming language agnostic:
- independent of the source ecosystem (Java, .NET, Python, TypeScript, ...)
- independent of its consumers, although an SBOM can be tailored towards a specific use case
What Is This Siemens Standard BOM?

The Siemens Standard BOM is a standardized SBOM format with tooling for Siemens.

It is

- a subset of OWASP CycloneDX
- programming language agnostic (It’s just JSON)
- independent of the source ecosystem (Java, .NET, Python, TypeScript, ...)
- independent of its consumers, although an SBOM can be tailored towards a specific use case (for example, it works with different Siemens software clearing toolchains)
Why Should We Have Standard BOM Rather Than Plain CycloneDX?

- Standard BOM is a proper subset of CycloneDX.
- SBOM components are presented in list form, not as a tree.
- Custom properties are not random free-form Strings as per CycloneDX, but elements from the Siemens Property Taxonomy for CycloneDX. CycloneDX reserves a siemens namespace for Standard BOM.
- Component Sources can be specified.
- A Standard BOM Package bundles the SBOM document with any referenced files, such as component sources or binary archives.

```
"properties": [ {
  "name": "siemens:direct",
  "value": "true"
}, {
  "name": "siemens:filename",
  "value": "commons-codec-1.13.jar"
}, {
  "name": "siemens:primaryLanguage",
  "value": "Java"
}, ... ]
```

```
"externalReferences": [ {
  "type": "distribution",
  "url": "file:sources/a1bdc0e8f_a35c23e197498d/log4j-api-2.11.2-sources.jar",
  "comment": "source archive (local copy)",
  "hashes": [ ... ]
}, {
  "type": "distribution",
  "url": "https://repo.maven.apache.org/maven2/.../log4j-api-2.11.2-sources.jar",
  "comment": "source archive (download location)",
  "hashes": [ ... ]
}, ... ]
```
Example:
BOM Entry for Java Library

```
{
  "type": "library",
  "author": "Henri Yandell <bayard@apache.org>, Tim OBrien ...",
  "group": "commons-codec",
  "name": "commons-codec",
  "version": "1.13",
  "purl": "pkg:maven/commons-codec/commons-codec@1.13?type=jar",
  "description": "The Apache Commons Codec package contains ...",
  "hashes": [ ... ],
  "licenses": [ {
    "license": {
      "name": "Apache License, Version 2.0",
      "url": "https://www.apache.org/licenses/LICENSE-2.0.txt"
    }
  }],
  "externalReferences": [ ... ],
  "properties": [ {
    "name": "siemens:direct",
    "value": "true"
  }, {
    "name": "siemens:primaryLanguage",
    "value": "Java"
  }, {
    "name": "siemens:thirdPartyNotices",
    "value": "Apache Commons Codec\nCopyright 2002-2019 The Apache Software Foundation\nThis product includes software developed at The Apache Software Foundation (https://www.apache.org/).\nsrc/test/org/apache/commons/code/language/DoubleMetaphoneTest.java\ncontains test ..."
  }],
  "copyright": "Copyright 2002-2019 The Apache Software ...",
  "bom-ref": "pkg:maven/commons-codec/commons-codec@1.13?type=jar"
}
```
Standard BOM is Great For Automated Pipelines Which Need SBOMs
Example: Software License Compliance
Road To Collaboration

We presented our experiences and our approach on SBOMs. We could imagine that some of our tools are upstream contributions or candidates for open-sourcing.

➔ Would you like us to open-source these tools?

➔ Some of our tools are from-scratch implementations of existing OSS tools – we do not want to compete, but rather collaborate with them

Based on https://geek-and-poke.com/
Siemens product portfolios are complex, therefore SBOM generation is also complex.

SBOMs are created with a particular use case in mind.

A common format facilitates internal collaboration on SBOM tooling.

Accurate and complete SBOMs are key.

SBOMs as a cross-cutting concern work better when developed and introduced collaboratively.

Containers prove particularly challenging.

CycloneDX is great, but from a consumption perspective, some things need to be more concrete.

SBOM are puzzles, you must combine many things to get an accurate SBOM.
Contact

Thomas Graf
Email thomas.graf@siemens.com
GitHub: https://github.com/tngraf

Alexander Gschrei
Email alexander.gschrei@siemens.com
GitHub: https://github.com/agschrei

Thomas Jensen
Email jensethomas@siemens.com
GitHub: https://github.com/tsjensen
BACKUP
The „Standard BOM Package“

Used for handling file system references in the SBOM.
Objective: Self-contained package

All external references used in the SBOM must be either
• URLs of publicly available resources on the Internet, or
• a relative file system path.

Resources referenced via relative paths become part of the self-contained "Standard BOM Package", which is a ZIP file or file system folder.

```json
sbom.json
├── binaries
│   ├── 77100a62c2e6f04b53977b9f541044d7d722693d
│   └── some-binary.jar
│         ├── 8031352b2bb0a49e67818bf04c027aa92e645d5c
│         └── another-binary.jar
│               └── (... more ...)
└── sources
    ├── 6bb10559db88828dac3627de26974035a5dd4dd
    └── some-binary-sources.jar
        └── 4d44e4edc4a7fb39f09b95b09f560a15976fa1ba
            └── another-binary-sources.jar
                └── (... more ...)
```