APH10

FOSDEM

A day in the life of a SBOM

February 2026

Anthony Harrison - anthony@aph10.com

Who am I?

40 years delivering mission critical solutions across multiple sectors

Founder and Director APH10

Co-founder SBOMEurope

Open Source Software

Mentor

An increasingly (maybe) common question…

Can you give me a SBOM for your product?

APH10

What’s the problem?

- What is the **use case** for the SBOM?

- Do you understand the different **formats** of SBOMs?
  - And versions?

- Do you know about the different **types** of SBOMs?

Understand SBOM Use Cases

- **Licence Compliance** to ensure that 3rd party components are being used in accordance with the requirements of the licence.
- **Vulnerability Assessment and Management** to support software security and resilience.
- **Support for Regulatory Compliance** by provision of artefacts required to support cyber security needs.
- **Support for ESCROW** by providing an inventory of the components used in the production of a software product.
- **Support to M&A** activities by providing transparency into the construction of a software product.

APH10

All SBOMs are not created equally

One SBOM is unlikely to meet all use cases

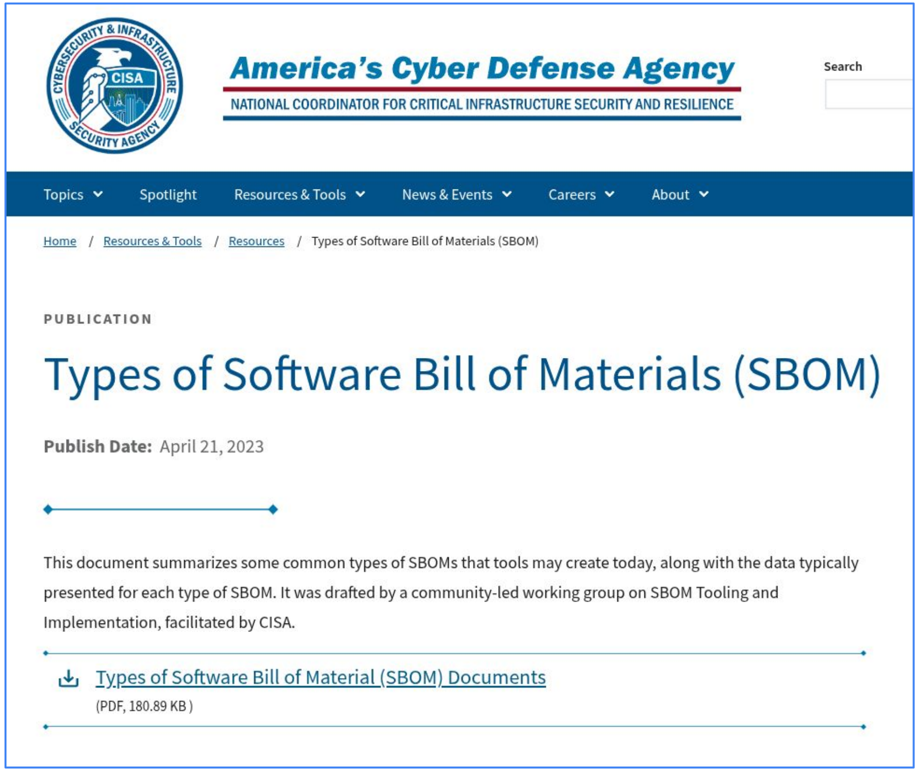
APH10

An increasingly (maybe) common question…

Can you give me *the right* SBOM for your product?

SBOM Types

- Design
- Source
- Build
- Analysed
- Deployed
- Runtime


America's Cyber Defense Agency
NATIONAL COORDINATOR FOR CRITICAL INFRASTRUCTURE SECURITY AND RESILIENCE
Search
Topics
Spotlight
Resources & Tools
News & Events
Careers
About

Home / Resources & Tools / Resources / Types of Software Bill of Materials (SBOM)

PUBLICATION

Types of Software Bill of Materials (SBOM)

Publish Date: April 21, 2023

This document summarizes some common types of SBOMs that tools may create today, along with the data typically presented for each type of SBOM. It was drafted by a community-led working group on SBOM Tooling and Implementation, facilitated by CISA.

Types of Software Bill of Material (SBOM) Documents
(PDF, 180.89 KB)

https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom

SBOM Type Support

- SPDX - Software profile ->SbomType
  - Available since version 3.0
- CycloneDX - Metadata->Lifecycle
  - Available since version 1.5

SBOM Type	SPDX	CycloneDX
Design	Design	design
Source	Source	pre-build
Build	Build	build
Analysed	Analysed	post-build
Deployed	Deployed	operations
Runtime	Runtime	discovery
		decommissioned

APH10

Identification of SBOM types

SEI PlugFest 2024 (https://github.com/cmu-sei/sbom-plugfest-2024/tree/main)

SPDX and CycloneDX SBOMs

- Source and Build

Only 1 vendor out of 9 SPDX submissions included sbomType

- The only one which used SPDX 3

Only 1 vendor out 11 of CycloneDX submissions included lifecycle element

- All but 1 submission were using version 1.5 or 1.6

EU CRA Requirements

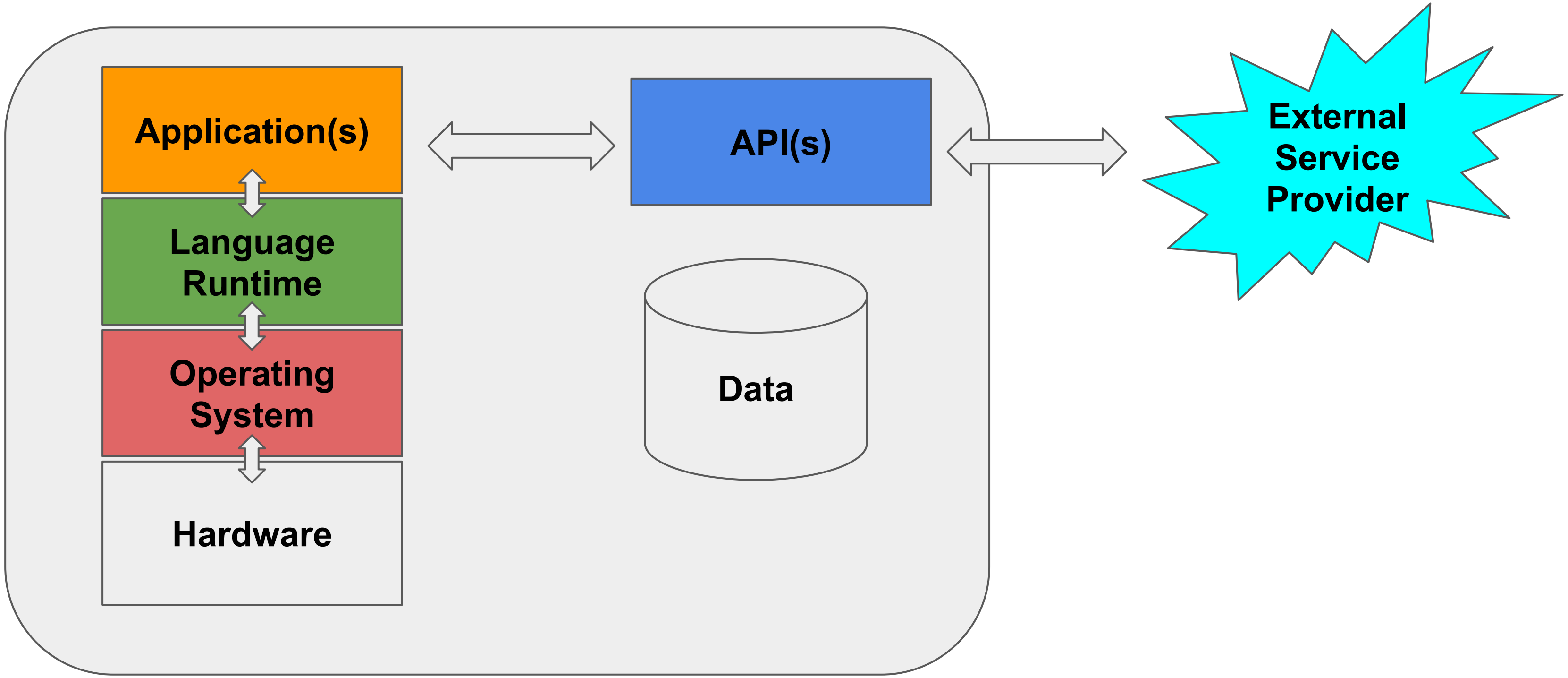
“(digital products) … be made available on the market without ***known exploitable*** *vulnerabilities”*

(CRA Annex I Part 1 2(a))

Which SBOM type is needed for the CRA?

APH10

A typical product



Design SBOMs

- SPDX - SBOM of intended, **planned software project** or product with included components (some of which may not yet exist) for a new software artifact.

- CycloneDX - BOM produced early in the development lifecycle containing inventory of components and services that are proposed or **planned to be used**. The inventory may need to be procured, retrieved, or resourced prior to use.

Design SBOM - Use cases

- **Early Risk Assessment**: Identify potential licensing conflicts or known vulnerabilities in planned components (although the version of a component will unlikely to be known unless this is an identified constraint) before development even begins.
- **Supplier Due Diligence:** Evaluate the security posture of third-party libraries and services you intend to integrate. Are they maintained? How quickly are defects, and in particular those related to security, remediated?
- **Architectural Compliance:** Ensure chosen components align with organisational security policies and architectural standards.
- **Cost Estimation:** Forecast potential licensing costs associated with open-source or commercial components.

APH10

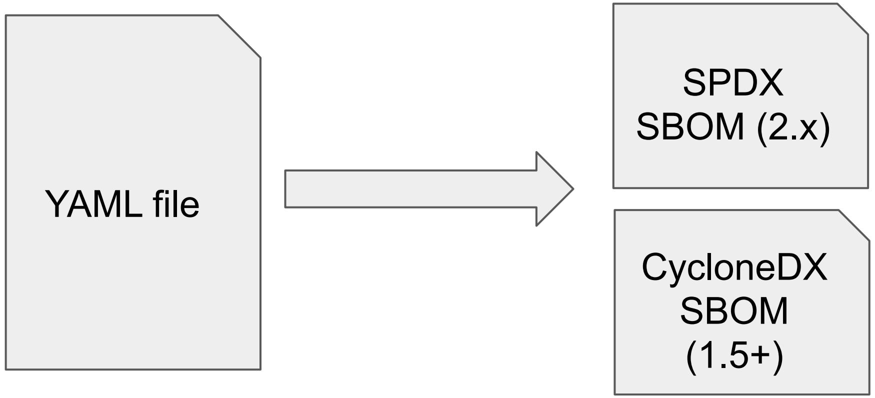
Design SBOM - Tooling

Couldn't find any 😐

So I created one! SBOMac (SBOM-as-code)

- Possible Components and environment

Nothing about the actual target environment



```yaml
# Example yaml file
name: Design SBOM
author: APH
email: aph@aph10.com
supplier: ACME Inc
element:
- name: Payment System
 type: system
 element:
   - name: Platform
     type: hardware
     element:
     - name: Network
       type: hardware
       description: 10GB
     - name: Memory
       type: hardware
       description: At least 64GB
   - name: OS
     type: operating_system
     description: An embedded OS
     element:
     - name: Database
       type: application
       description: A SQL database
       product: SQLlite
       comment: Version should be at least 3.0
     - name: Awesome Application
       type: software
       description: A payment processor
```

Source SBOMs

- SPDX - SBOM created directly from the development environment, **source files**, and included dependencies used to build an product artifact.

- CycloneDX - BOM consisting of information obtained prior to a build process and may contain **source files** and development artifacts and manifests. The inventory may need to be resolved and retrieved prior to use.

Source SBOMS - Use cases

- **Developer Visibility:** Provide developers with a clear view of their direct dependencies. Depending on how the dependencies are specified (are pinned versions specified?), specific versions of the dependencies may not be known.
- **Licence Compliance (Early)**: Identify and manage licences of declared dependencies at the source level.
- **Vulnerability Scanning (Early)**: Perform initial vulnerability scans based on declared dependencies.
- **Code Review Enhancement:** Aid in reviewing dependencies brought in by new features or modules.

Source SBOMS - Tooling

Lots of tools

- Licences
- Copyright
- Direct Dependencies

Nothing about the actual target environment

```
  "type": "file",
      "bom-ref": "3-shared_functions",
      "name": "shared_functions.py",
      "hashes": [
        {
          "alg": "SHA-1",
          "content": "8246e962f691b00cc2cec2d7b4c073f3e4dafa55"
        },
        {
          "alg": "SHA-256",
          "content":
"274b63f8c1ce8359d8cb6aff9706942f94439c6291edc3df408c4d773513962d"
        },
        {
          "alg": "SHA-512",
          "content":
"e7684cac81ca1d416f448d246d9b9b32d9e43e6a814f321341a48f3686cd0ce87a74c3b535910760988
f14abd7c854c1e0c68dd679ee787276d931b3292a04a2"
        }
      ],
      "copyright": "2025 Carnegie Mellon University.",
      "properties": [
        {
          "name": "Comment",
          "value": "Source is Python"
        }
      ]
```

Build SBOMs

- SPDX - 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.

- CycloneDX - BOM consisting of information obtained during a **build process** where component inventory is available for use. The precise versions of resolved components are usually available at this time as well as the provenance of where the components were retrieved from.

Build SBOM - Use Cases

- **Supply Chain Integrity**: Verify that only approved and expected components are included in the build.
- **Reproducible Builds**: Essential for ensuring that a given source code always produces the same binary or product, a critical step for trust.
- **Attestation**: Create cryptographic attestations about the build process and included components.
- **Early Vulnerability Remediation**: Pinpoint exact versions of vulnerable components that made it into the build, allowing for precise patching.

Build SBOMs - Tooling

Lots of tools - Often operating on manifests/package managers

- Transitive Dependencies
- Component Versions
- Component Identifiers
- Component ecosystem information
- More Licences

Nothing about the actual target environment

Build SBOMs - Tooling

```
{
      "type": "library",
      "bom-ref": "7-jsonschema",
      "name": "jsonschema",
      "version": "4.17.3",
      "supplier": {
        "name": "Julian Berman",
        "contact": [{"email": "Julian+jsonschema@GrayVines.com"}]
      },
      "cpe": "cpe:2.3:a:julian_berman:jsonschema:4.17.3:*:*:*:*:*:*:*",
      "purl": "pkg:pypi/jsonschema@4.17.3",
      "description": "An implementation of JSON Schema validation for Python",
      "hashes": [
        {
          "alg": "SHA-256",
          "content":
"a870ad254da1a8ca84b6a2905cac29d265f805acc57af304784962a2aa6508f6"
        }
      ],
      "licenses": [
        {
          "license": {
            "id": "MIT",
            "url": "https://opensource.org/license/mit/",
            "acknowledgement": "concluded"
          }
        }
      ],
```

```
"externalReferences": [
        {
          "url": "https://github.com/python-jsonschema/jsonschema",
          "type": "website",
          "comment": "Home page for project"
        },
        {
          "url": "https://pypi.org/project/jsonschema/4.17.3/#files",
          "type": "distribution",
          "comment": "Download location for component"
        },
        {"url": "https://python-jsonschema.readthedocs.io/","type": "documentation"},
        {"url": "https://github.com/python-jsonschema/jsonschema/issues/","type": "issue-tracker" },
        {"url": "https://github.com/python-jsonschema/jsonschema/blob/main/CHANGELOG.rst","type": "log"},
        {"url": "https://github.com/python-jsonschema/jsonschema","type": "vcs"}
      ],
      "properties": [
        {
          "name": "release_date",
          "value": "2022-11-29T20:37:45Z"
        },
        {
          "name": "language",
          "value": "Python"
        },
        {
          "name": "python_version",
          "value": "3.10.8"
        }
      ]
    },
```

APH10

Analysed SBOMs

- SPDX - 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.

- CycloneDX - BOM consisting of information obtained after a build process has completed and the resulting components(s) are available for further **analysis**. Built components may exist as the result of a CI/CD process, may have been installed or deployed to a system or device, and may need to be retrieved or extracted from the system or device.

Analysed SBOMs - Use Cases

- **Discovering Hidden Dependencies**: Uncover components that weren’t explicitly declared.
- **Deep Vulnerability Analysis**: More accurately assess vulnerabilities by understanding the call graph and reachability of vulnerable functions.
- **Malware Detection**: Identify malicious or unauthorised components embedded within the software.
- **Compliance Auditing:** Provide a more robust and verifiable list of components for regulatory compliance.

Analysed SBOMs - Tooling

Do we need it?

- Can analyse multiple SBOMs to find common components etc
- Can analyse SBOMs for deficiencies (missing licences, non-compliant licencies, obsolete or vulnerable components)

Deployed SBOMs

- SPDX - 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.

- CycloneDX - BOM produced that represents inventory that is running and operational. This may include staging or **production environments** and will generally encompass multiple SBOMs describing the applications and operating system, along with HBOMs describing the hardware that makes up the system. Operations Bill of Materials (OBOM) can provide full-stack inventory of runtime environments, configurations, and additional dependencies.

Deployed SBOMs - Use Cases

- **Deployment Security Baseline**: Establish a baseline of components for production environments.
- **Incident Response Preparation**: Quickly identify components in deployed software during a security incide**nt.**
- **Vulnerability Management (Late)**: Perform vulnerability scans based on what's actually installed and prioritise patching efforts based on an assessment of risk.
- **Licence Compliance (Late)**: Identify and manage licences of all dependencies.
- **Configuration Drift Detection**: Compare deployed SBOMs over time to detect unauthorised changes.

Deployed SBOMs - Tooling

Some tools

- Precise target definition and configuration
- Runtime dependencies (dynamic linkages)
- Every component has an identifiable version
- Every component has a licence (or should!)
- Every component has an identifier

Defines the actual target environment which the product operates in

<u>These are the components to be monitored for exploits</u>

Runtime SBOMs

- SPDX - SBOM 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.

- CycloneDX - BOM consisting of **information observed** through network discovery providing point-in-time enumeration of embedded, on-premise, and cloud-native services such as server applications, connected devices, microservices, and serverless functions

Runtime SBOMs - Use cases

- **Real-time Vulnerability Monitoring:** Detect and alert on vulnerabilities in components actively being used in production.
- **Behavioral Anomaly Detection**: Identify unexpected or unauthorised components being loaded or executed.
- **Supply Chain Attack Detection**: Pinpoint malicious components introduced or activated during runtime.
- **Optimised Resource Allocation**: Understand which components are truly necessary for a given workload.

Runtime SBOMs - Tooling

Not found 😐

So I am working on one…. sbom4runtime

Benefit is it identifies what is actually used (at a moment in time)

- May help reduce attack surface by identifying unused components and revising the deployed SBOM

Runtime SBOMs - Tooling

```
{
      "type": "library",
      "bom-ref": "2-libc.so.6",
      "name": "libc.so.6",
      "evidence": {
        "occurrences": [
          {
            "location": "/usr/lib/x86_64-linux-gnu/libc.so.6"
          }
        ]
      },
      "properties": [
        {
          "name": "library_loaded",
          "value": true
        }
      ]
    },
```

To be added

- Network connections
- Files

Summary

	Design	Source	Build	Analysed	Deployed	Runtime
Licence Compliance		Partial	Partial		Fully	
Vulnerability Management		Limited	Partial		Fully	
Regulatory Compliance		Unlikely	Unlikely		Probably	
Other	Depends	Dependes	Depends	Depends	Depends	Depends

	Design	Source	Build	Analysed	Deployed	Runtime
lib4sbom	Y	Y	Y	Y	Y	Y
sbomac	Y	Y				
sbom4files		Y				
sbom4python		Y	Y			
sbom4js			Y			
sbom4php			Y			
sbom4rust			Y			
sbom4windows					Y	
distro2sbom					Y	
sbom4runtime						Y

Apache-2.0 Licence. Work with both CycloneDX and SPDX

Summary

- Still a lot of confusion when asking for ‘a SBOM’
  - Use case determines when in lifecycle SBOM should be produced
  - Most useful information isn’t confirmed until late in the development lifecycle
- Lack of awareness of different types of SBOM
  - Most tools only support one type
  - Very limited tool support in identifying SBOM type
- Do we have too many types?
  - How useful are the different types in supporting CRA?
- Do we have the right types?

Contact Details - Anthony Harrison

LinkedIn

GitHub

https://www.linkedin.com/company/aph10/
https://www.linkedin.com/in/anthonypharrison/

Email: anthony@aph10.com

https://www.aph10.com
https://www.sbomeurope.eu
https://www.sbomfocus.eu