Software Bill of Materials: SBoM

Author: Robert Marion

Preface:

This Document was originally prepared in late 2022 to early 2023. SBoM tooling is rapidly changing and this document reflects an accurate assessment at the time it was created. Updates can be tracked via GitHub.

Abstract:
Software supply chain issues have come to the forefront of the security world's consciousness ever since the Solar Winds hack of 2020. As a direct consequence, a United States Presidential Executive Order (excerpted below) and additional pending litigation, not only in the US but in Europe (the Cyber Resilience Act) have mandated better accountability into determining what components comprise a software product. This document demonstrates some practical strategies to implementing a Software Bill of Materials (SBoM).

At the time of its conception, the practical implementation of an SBoM had not been fully considered. One serious concern is the speed with which third party components change and the introduction of newly found vulnerabilities in those components.

SBoM Definitions, Tools and Automation

Definition

- A SBoM documents the composition of a software product, library, or component
    - SBoMs were mandated by Executive Order 14028[1] in response to software supply chain attacks.
        - SBoMs allow a consumer to determine whether a component within a library has a known vulnerability
        - Shows the provenance of 3rd party component
        - Contains licensing information, assuring a consumer that the product they obtain does not violate licensing issues
    - It must include COTS in addition to FOSS.
- From Executive Order 14028:
    - 4 Enhancing Software Supply Chain Security.
    - (F) monitoring operations and alerts and responding to attempted and actual cyber incidents;
    - (vii) **providing a purchaser a Software Bill of Materials (SBOM) for each product directly or by publishing it on a public website**;
    - the National Telecommunications and Information Administration, shall publish minimum elements for an SBOM (see below)

Sec 10. Definitions (j) *the term "Software Bill of Materials" or "SBOM" means a formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open source and commercial software components. The SBOM enumerates these components in a product. It is analogous to a list of ingredients on food packaging. An SBOM is useful to those who develop or manufacture software, those who select or purchase software, and those who operate software. Developers often use available open source and third-party software components to create a product; an SBOM allows the builder to make sure those components are up to date and to respond quickly to new vulnerabilities. Buyers can use an SBOM to perform vulnerability or license analysis, both of which can be used to evaluate risk in a product. Those who operate software can use SBOMs to quickly and easily determine whether they are at potential risk of a newly discovered vulnerability. A widely used, machine-readable SBOM format allows for greater benefits through automation and tool integration. The SBOMs gain greater value when collectively stored in a repository that can be easily queried by other applications and systems. Understanding the supply chain of software, obtaining an SBOM, and using it to analyze known vulnerabilities are crucial in managing risk.*[1]

Requirements

- Who can request an SBoM?
    - All purchasers. EO 14028 Sec 4. e. vii
- An SBoM should reflect the current state of a piece of software.[3]

NTIA Formats

NTIA identifies three formats in widespread use[3,9]

- **SPDX**: Originally used for licensing but has grown in scope.
    - SPDX 2.1 has been the most commonly used version. In May of 2023, SPDX 3.0 has been released. This is significant because it is designed to address both the NTIA SBOM requirement and the Cyber Resiliency Act.
- **CycloneDX**: a lightweight SBOM standard that originated with OWASP
- **SWID**: primarily used by commercial software publishers. Not relevant to this analysis.

The two competing formats (SPDX and CycloneDX) are likely to stay. Of the SBOM tools tested, Syft produces both formats.

Potential Automation Solutions for Generating SBOMs:

- Mend (previously known as WhiteSource) Renovate: https://www.mend.io/sbom/
    - This is a commercial SCA (Source Composition Analysis) tool
    - The WhiteSource sbom generator works by turning a WhiteSource inventory report into SPDX format. This has some disadvantages:
        - assumes everything scanned is inventory
        - assumes that all software materials are scanned with WhiteSource
        - will require some experimentation to catch all language types
- Snyk
    - This is also a commercial SCA tool
    - Snyk has SBoM functionality in beta.
    - Snyk does a very good job at identifying 3rd party dependencies.
    - Because it does a poor job at 1st party dependencies, this solves the problem of differentiating between 1st and 3rd party deps
    - Results currently exported in CycloneDX but will also be exported to SPDX format.
    - Does not currently handle C/C++ or Docker images (yet).
    - Does not (yet) allow for excluding directories or particular files.
- Microsoft: https://devblogs.microsoft.com/engineering-at-microsoft/generating-software-bills-of-materials-sboms-with-spdx-at-microsoft/
    - Microsoft has an *open-source* tool that should be platform agnostic
        - sbom-tool is able to generate an sbom for C/C++ projects if it is using Cmake
        - will do a build, when possible, and catches more transitive dependencies
        - Can scan Docker images (not tested)
        - It does not lock a user into a particular product, such as Snyk or WhiteSource/Mend. If you drop a vendor, you still have a solution.

        - It only outputs in SPDX format (no CycloneDX support)
        - It is a very new tool and has some bugs (ex: in version 0.3.0 the generator fails in Windows but works in Linux)
        - For more info, see: https://github.com/microsoft/sbom-tool and Here's a neat intro video: https://www.youtube.com/watch?v=ZRZUZG9vg7k
    - But:
        - Initial testing looks good. A deeper PoC is forthcoming.
- Syft https://github.com/anchore/syft
    - Recommendation appears in https://mergebase.com/blog/best-tools-for-generating-sbom/
    - Initial testing produces a usable SBOM
    - Does **not** do a build and does not catch transitive dependencies.
    - Can output in SPDX or CycloneDX
    - Open source project by Anchore. Paid support exists.
- Rezilion https://www.rezilion.com/
    - This is a commercial product that looks promising. Not evaluated (yet).

SBOM Generation Tool Comparison

Tool	Tested Version	Current Version	Format(s)	C++	Docker	Issues
Microsoft	0.3.1	1.1.1	SPDX 2.2	Cmake	Yes	• SPDX format is not as useful as CycloneDX because the converters do not have purls
Snyk	Beta		CycloneDX	No	No	• Must maintain contract to continue using this. • Creates one SBOM per component, which means a single repo could have hundreds of SBOMs. Apply CycloneDX CLI **merge** to combine SBOMS for a product.
Syft	0.64.0	0.80.0	CycloneDX SPDX	Conan	Yes	• No REST API but uses an optional yaml file • Use of Conan is not as widespread as cmake.

Some Useful Open Source Tools

CycloneDX CLI

CycloneDX CLI is a very useful (possibly necessary) set of utilities that: **convert**s SPDX to CycloneDX format, **merge**s two or more SBOMs, **validate** a BOM, etc. *This tool is still in development and testing has shown that it has some bugs.*

- Tested: version 0.24.2 (Oct 12, 2022)
- **Convert** example from SPDXJSON to CycloneDX JSON format:
  - cyclonedx-win-x64.exe convert --input-file sample.spdx.json --input-format spdxjson --output-format json --output-file output.json
- **Merge** is needed to combine two or more BOMs. If SBOMs are generated by Snyk, for example, a single repository will create an SBOM for each project. This is undesirable because an SBOM needs to express the contents of a product.
  - Ex: cyclonedx-win-x64.exe merge --input-files sbom1.json sbom2.json --output-file sbom_all.json

⚠**Caveat:** CycloneDX CLI (v 0.24.2)) will fail to convert or validate a BOM generated by the Microsoft sbom-tool (as of version 0.3.1). The hack to

fix this is to replace "PACKAGE-MANAGER" in the "externalRefs" section with "PACKAGE_MANAGER" (i.e., replace the dash with an

underscore throughout the document). ⚠

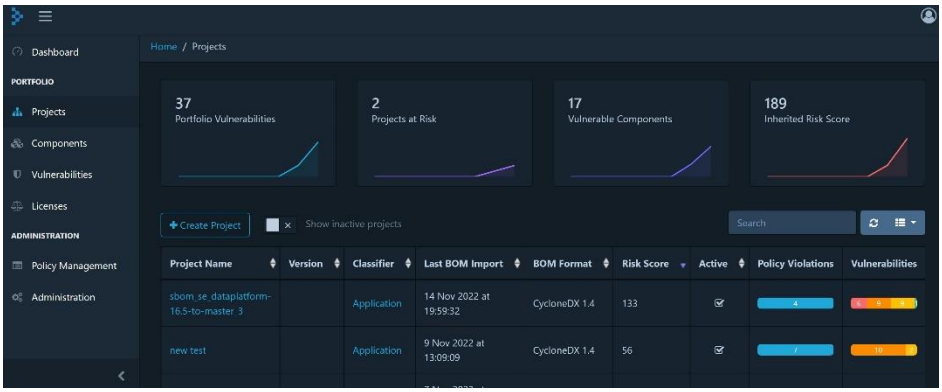
Dependency Track by OWASP

Dependency Track by OWASP for consumption of an SBOM.

Dependency Track lets you upload SBOMs in CycloneDX format. It will display the components, licenses and even look up vulnerabilities based on their PURL (project urls).

- The one tool to rule them all is Dependency Track by OWASP.
- Initial testing of Dependency Track looks very good.
- It's API first design should make it a great candidate for CI pipeline integration or any other automation.
- Dependency Track runs as a Docker image.
- It includes VEX capability (not yet tested)
- And it is free, open source!

The only drawback to this tool is that it only works with CycloneDX format and so a conversion tool is required if you want to examine/consume something in SPDX format. Converting, given current tools available, will leave out PURLs. In these cases, vulnerability information will not be displayed.



Dependency Track

Best Approaches for SBOM Generation: Snyk or Microsoft SBOM

Requirements:

- Must work in the CI pipeline and be easily automated.
- SBOM generation should be triggered as part of a release build.
    - Common automation servers: Jenkins, TeamCity, Git …
- It should also be runnable as a stand-alone script from anywhere.
- Must be in either SPDX or CycloneDX format and preferably can generate both.

Snyk

Beta is limited so far. It has been tested and works reasonably well given limitations.

Automation should be easy.

Limitations:

- Will not work for C/C++ projects (currently)
- By not running an Open Source solution, you risk losing this functionality if you drop Snyk as a vendor
- Does not currently work (as of May 2023) for container scanning
- Only produces CycloneDX (not much of a problem)

Microsoft SBOM Tool

- As an open source tool, this removes a vendor requirement.
- Will analyze cmake files for C/C++ code.
- Only produces SPDX. The OWASP CycloneDX CLI converted SBOMs are correct but converted SBOMs do not yield vulnerability information when imported into Dependency Track.
- Automation for the CI pipeline appears to be something that can be reasonably achieved.

Syft

- As an open source tool, this removes a vendor requirement
- Output can be configured to be SPDX or CycloneDX
- Has no API but uses a configurable yaml file
- Allows for exclusions of directories and files
- Will analyze Conan for C/C++
- TODO more testing. Determine if it misses any of the package-managed source and test C++

VEX: Vulnerability Exploitability eXchange

> The goal [of a VEX] is to communicate the exploitability of components with known vulnerabilities in the context of the product in which they are used.[11]

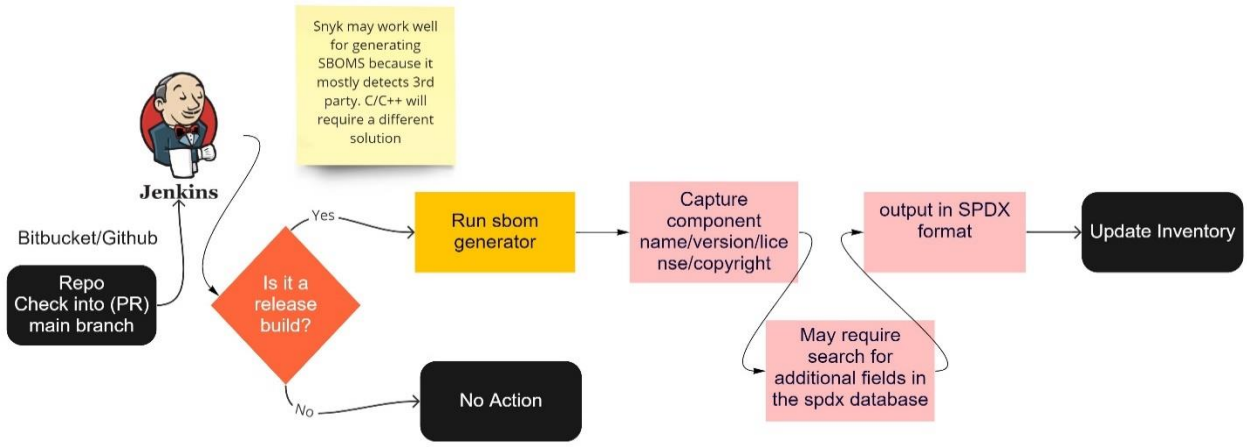
SBOMs are relatively static but the vulnerabilities in the components of an SBOM will be very dynamic, a VEX:

- Should, be a separate document from an SBOM (it is inadvisable to create one, integrated document).
- Integrating an SBOM with its VEX BOM is vendor and format dependent.
- As mentioned above, Dependency Track by OWASP may be a good solution to the problem.

Just like SBOM generators and consumers, there are many VEX solutions. OpenVex is, Apache licensed and is worth exploring (TBD).

A note on VEX. VEX is not a silver bullet for dealing with the speed of change that can quickly change an SBOM with a “clean bill of health” to one which has severe vulnerabilities. A VEX will likely require a dependency on either development engineers or security engineers to declare that a particular component with a vulnerability is not exploitable in the context in which it is used. The author of this document is still considering methods for automation.

SBOM (Tool Agnostic) PoC



Testing Matrix

Microsoft SBOM tool: SBOM Generation

Steps:

1. Select repos for testing
2. Generate SBOMS using: sbom-tool-win-x64.exe generate ...
    1. Ex: >sbom-tool-win-x64.exe generate -b ./outputdir -bc C:/SBoM/test_data/data_set_1 -pn MyTest -pv 1.0 -nsb https://www.somepath.com -ps SomeName
3. Convert from SPDXJSON format to CycloneDX JSON format.
    1. Do this using the CycloneDX CLI tool with the "convert option"
    2. Ex: cyclonedx-win-x64.exe convert --input-file sample.spdx.json --input-format spdxjson --output-format json --output-file output.json
4. Make the CycloneDX human readable by scanning it with Dependency Track (discussed above).
5. Verify results for accuracy and completeness. Include testing for C/C++ and container files and note results.

Important Note Regarding Data Used for Testing: This chart is being revised for two reasons: 1. Will use updated versions of tools for testing, 2. I discovered the data sets I was using contained some proprietary code and therefore I cannot release the data sets.

	Input repo(s)	Repo size/file count	Composition	Components discovered	Errors: components not found	C++/Docker [TODO]	Issues/Not

Snyk	DataSet_1	20.9 MB/3,015 files	Maven/Java/Python	1393	TODO		Pending verificatior
Microsoft	DataSet_1	TODO					
Syft	DataSet_1			689			
Snyk	DataSet_2						
Microsoft	DataSet_2						
Syft	DataSet_2	105					Did not fin C/C++ because there were no Conan files

Pipeline Considerations

An important consideration when attempting to build an SBOM in the CI pipeline is whether the code is production code. If it is not, you are very unlikely to want that code to be included in your report. There are multiple ways of achieving this but two possibilities are:

1. Prior to any SBOM generation, designate certain repos/branches as production only. They may not include test code, PoC code, etc.
2. Create an exclusion/inclusion list of repos and allow for drilling down to directory levels. A repo may be production code but may also include testing directories or PoC code that will never be "shipped"

Regardless of which technology or method you use to build the SBoM, one way to designate whether code is production, test, or whatever is to create a pipeline.yaml file that is placed in each repo and which designates the intended usage of the code.

Ex:

```
sbom:
    enable: true

    branch: master

    exclusions: {list of directories}
```

Conclusion

All the tools that have been tested are somewhat new and each has their advantages and disadvantages. As time goes by, these should all get better. Some promising solutions may be commercial, such as Snyk, or Mend and some may be Open Source such as the OWASP tools, Syft or the Microsoft SBOM tool for generating SBOMS. OWASP Dependency Track does a very good job for consuming SBOMS and it is well worth the time to learn and experiment with it. That tool would be better if it could accept SPDX format in addition to CycloneDX. Remember, it is an active open source project so, if you want to improve the functionality, you can make a suggestion in the Discussions section of their GitHub page or contribute to the project.

C/C++ remains an issue. Some coverage is possible with the Microsoft tool if the code has cmake files and coverage is possible with Syft if the Conan package manager is being used. The Snyk tool, at present does not handle C/C++ but work is being done on that.

✅Syft almost does everything needed. It may omit transient dependencies, which might actually be acceptable. C/C++ scanning omits cmake. A hybrid approach where C/C++ is scanned with the Microsoft tool and everything else is scanned with Syft and the two SBOMs are combined using the CycloneDX **merge** function may be a very good way to build an SBOM that meets customer requirements.

Or

✅The Microsoft SBOM tool will build, catch transitive dependencies, and use cmake to catch more C/C++ than Conan. It is locked into SPDX format, which is perfectly acceptable as it meets requirements. Unfortunately, reading files in Dependency Track will require converting to CycloneDX with either the Syft or CycloneDX CLI tool but it impedes using Dependency Track's vulnerability discovery since the conversion does not create PURLs for components.

References

1. Executive Order 14028: https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity
2. National Telecommunications and Information Administration: https://www.ntia.gov/SBOM
3. SBoM Formats: https://www.ntia.gov/files/ntia/publications/ntia_sbom_formats_and_standards_whitepaper_-_version_20191025.pdf
4. WhiteSource: https://www.whitesourcesoftware.com/sbom/
5. WhiteSource: https://www.whitesourcesoftware.com/wp-content/media/2021/10/The-Importance-Of-SBOMs-In-Protecting-The-Software-Supply-Chain.pdf
6. Microsoft's approach: https://devblogs.microsoft.com/engineering-at-microsoft/generating-software-bills-of-materials-sboms-with-spdx-at-microsoft/
7. CycloneDX: https://cyclonedx.org/ and https://cyclonedx.org/tool-center/
8. Microsoft's Salus: https://github.com/microsoft/sbom-tool
9. SBoM format comparison: https://www.settletop.com/insights/understanding-sbom-standards-cyclonedx-spdx-swid

10. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM): Second Edition: https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf
11. VEX: https://cyclonedx.org/capabilities/vex/