Carnegie Mellon University
Software Engineering Institute

SOFTWARE BILL OF MATERIALS (SBOM) CONSIDERATIONS FOR OPERATIONAL TEST & EVALUATION ACTIVITIES

Michael Bandor

June 2024

Overview and History of Software Bill of Materials

The Software Engineering Institute (SEI) was tasked to look at the current state of Software Bill of Materials (SBOM) and the potential applicability for Operational Test & Evaluation activities. As part of the task, the following sources were considered in the development of the use cases and supporting questions/challenges:

- *Cybersecurity OT&E Guidance*
- *Cybersecurity Test and Evaluation Guidebook,* V2.0*,* 10 Feb 2020, Change 1
- *Cybersecurity Test and Evaluation Guidebook, For Official Use Only Appendices*, Version 2.0, 30 June 2018
- *Cybersecurity Test and Evaluation Guidebook Addendum, Cybersecurity Test and Evaluation of Department of Defense Systems Hosted on Commercial Cloud Service Offerings*, V1.0, Dec 2019
- *Cyber Economic Vulnerability Assessments (CEVA) Memorandum*, 21 Jan 2015
- *Procedures for Operational Test and Evaluation of Cybersecurity in Acquisition Programs Memorandum*, 3 Apr 2018
- *Full-Spectrum Survivability and Lethality, Operational and Live Fire Test & Evaluation*, DoD Manual 5000.89C
- *Realistic Full Spectrum Survivability and Lethality Testing*, DoD Manual 5000.UT
- *Cyber Development Test and Evaluation*, DoD Manual 5000.UY
- *Operational and Live Fire Test and Evaluation of Software*, DoD Manual 5000.96
- *Test and Evaluation Enterprise Guidebook*, Aug 2022

Bottom Line & Recommendation

SBOMs, at this point in time, are in early and varying stages of adoption across industry and within the DoD. There are still issues with the quality (e.g., completeness, accuracy, currency, etc.) of the

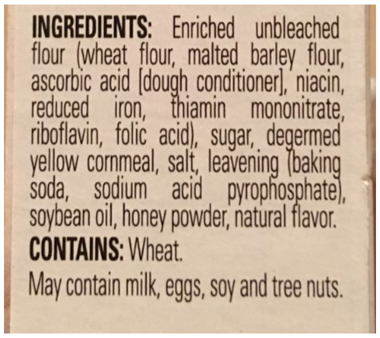
SBOMs being produced as well as adherence to the minimum essential elements identified by the US Department of Commerce. Legacy systems as well as cloud-based systems will present a challenge for producing SBOMs. The DoD is still working on proposed guidance for addressing the SBOM requirement by programs. In the absence of definitive guidance, some programs are proceeding on their own while others are taking more of a "wait and see" approach (treating the requirement as an "unfunded mandate").

Given this early phase of adoption, it is recommended that SBOMs be used to augment but not replace the current methods used by Operational Test (OT) personnel in performance of the testing functions and not to rely solely on the SBOM information. As the quality issues as well as widespread adoption become more prevalent over time, SBOMs will prove to be more useful for OT activities.

What is an SBOM?

The US Department of Commerce (DOC) defines an SBOM as follows [DOC 2021]:

> ***An SBOM*** *is a formal record containing the details and supply chain relationships of various components used in building software. In addition to establishing these minimum elements*[1]*, this report defines the scope of how to think about minimum elements, describes SBOM use cases for greater transparency in the software supply chain, and lays out options for future evolution.*



INGREDIENTS: Enriched unbleached flour (wheat flour, malted barley flour, ascorbic acid [dough conditioner], niacin, reduced iron, thiamin mononitrate, riboflavin, folic acid), sugar, degermed yellow cornmeal, salt, leavening (baking soda, sodium acid pyrophosphate), soybean oil, honey powder, natural flavor.
CONTAINS: Wheat.
May contain milk, eggs, soy and tree nuts.

An SBOM can be considered to be analogous to food packaging labels that identify the ingredients in the product. Where the analogy differs is the depth of information the SBOM looks at (e.g., dependencies, origins, etc.).

An SBOM can provide some useful information on its own, but for a more complete perspective additional information may be required. See Figure 1 for examples.

[1] The minimum elements, as identified in the US DOC guidance are **Author Name, Timestamp, Supplier Name, Component Name, Version String, Unique Identifier, and Relationship**. Additional information is discussed in detail later in this paper.

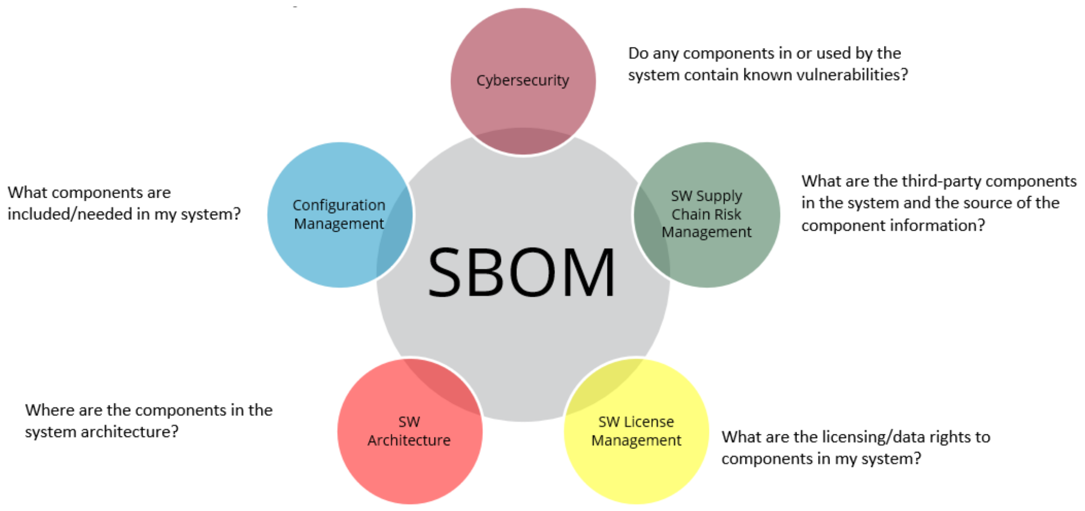

Cybersecurity
Do any components in or used by the system contain known vulnerabilities?
Configuration Management
What components are included/needed in my system?
SW Supply Chain Risk Management
What are the third-party components in the system and the source of the component information?
SBOM
SW Architecture
Where are the components in the system architecture?
SW License Management
What are the licensing/data rights to components in my system?

Figure 1: SBOM relationships to other areas

History of SBOMs

The concept of an SBOM is not a new one but rather dates to the 1990s and open-source software develop efforts. The current SBOM efforts and mandate are a result of Executive Order (EO)14028, *Executive Order on Improving the Nation's Cybersecurity*[2], issued 12 May 2021. In the EO, Section 4, *Enhancing Software Supply Chain Security*, it discussed the need for an SBOM and for secure software development efforts [EO 14028]. SBOMs are also a requirement of NIST SP 800-218, *Secure Software Development Framework (SSDF), Version 1.1*[3] also a result of EO 14028.

A brief timeline of the history of the SBOM efforts is shown in Table 1.

Table 1: Evolution of the SBOM (adapted) [4]

TIMELINE	EVOLUTION OF SBOM
In the 1990s	• Open-source software development led to the need for tracking and documenting dependencies.
In the 2000s	• More formalized practices were established to manage dependencies and licenses as the adoption of open-source software grew.

[2] https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[3] https://csrc.nist.gov/projects/ssdf

[4] Adapted from https://www.appknox.com/resources/guides/demystifying-source-code-v/s-binary-based-sboms-a-guide

TIMELINE	EVOLUTION OF SBOM
In the 2010s	• Software Bill of Materials (SBOMs) gained prominence in the context of software supply chain security, driven by concerns about software vulnerabilities and license compliance.
In 2016	• NIST highlighted the significance of SBOMs in its *Improving Software Supply Chain Security* report. This report paved the way for more widespread adoption of SBOMs
In 2018	• The National Telecommunications and Information Administration (NTIA) launches the *Multistakeholder Process on Software Component Transparency* to formulate and establish an SBOM including common, consensus definitions, and emphasis on a "baseline" SBOM.[5]
In the 2020s	• Binary-based SBOM gained momentum by adopting SBOM requirements in various industry standards and regulations.
In 2021	• Executive Order 14028, *Improving the Nation's Cybersecurity*[6] issued, directing the Secretary of Commerce (US Department of Commerce – US DOC), in coordination with the Assistant Secretary for Communications and Information, and the Administrator of the NTIA to publish minimum elements for an SBOM. • The *Minimum Essential Elements for a Software Bill of Materials (SBOM)*[7] issued by the US DOC and NIST • NTIA publishes *Software Suppliers Playbook: SBOM Production and Provision*[8]
In 2022	• OMB issues *Enhancing the Security of the Software Supply Chain through Secure Software Development Practices* memorandum (OMB M-22-18)[9], which addresses SBOMs as part of the practices • *Secure Software Development Framework (SSDF) V1.1*[10] issued with SBOM requirement (PS.3.2) and mapping to EO 14028
In 2023	• SEI publishes the *Software Bill of Materials Framework: Leveraging SBOMs for Risk Reduction*[11]

[5] https://www.ntia.gov/blog/marking-conclusion-ntia-s-sbom-process

[6] https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[7] https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

[8] https://www.ntia.gov/sites/default/files/publications/software_suppliers_sbom_production_and_provision_-_final_0.pdf

[9] https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf

[10] https://csrc.nist.gov/projects/ssdf

[11] https://insights.sei.cmu.edu/library/software-bill-of-materials-framework-leveraging-sboms-for-risk-reduction/

TIMELINE	EVOLUTION OF SBOM
	• The DoD, GSA, and NITA sponsor a proposed change to the Federal Acquisition Regulations (FAR-2021-0017) with SBOM-specific language to be added as part of the proposed changes.[12] • OMB issues an update (OMB-23-16)[13] to the original guidance (OMB-22-18) with clarifications on 1) third-party components, 2) freely obtained and publicly available proprietary software, and 3) federal contractor developed software. • NSA publishes *Securing the Software Supply Chain: Recommended Practices for Managing Open-Source Software and Software Bill of Materials*[14]
In 2024	• NSA publishes *Recommendations for Software Bill of Materials (SBOM) Management*[15] • CISA issues *Guidance on Assembling a Group of Products*[16] • The DoD officially releases the Data Item Description (DID) for SBOMs[17] and Hardware Bill of Materials (HBOMs)[18] for use as part of their Cybersecurity Supply Chain Risk Management efforts • CISA releases *Software Transparency in SaaS (Software as a Service) Environments* to discuss the value of SBOM-driven transparency in those environments and providing recommendations for advancing transparency[19]

Types of SBOMs

According to CISA [CISA 2023a], there are different types of SBOMs that can possibly be created today. The SBOM types are:[20]:

[12] https://www.federalregister.gov/documents/2023/10/03/2023-21328/federal-acquisition-regulation-cyber-threat-and-incident-reporting-and-information-sharing

[13] https://www.whitehouse.gov/wp-content/uploads/2023/06/M-23-16-Update-to-M-22-18-Enhancing-Software-Security-1.pdf

[14] https://media.defense.gov/2023/Dec/11/2003355557/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN%20RECOMMENDED%20PRACTICES%20FOR%20MANAGING%20OPEN%20SOURCE%20SOFTWARE%20AND%20SOFTWARE%20BILL%20OF%20MATERIALS.PDF

[15] https://media.defense.gov/2023/Dec/14/2003359097/-1/-1/0/CSI-SCRM-SBOM-MANAGEMENT.PDF

[16] https://www.cisa.gov/sites/default/files/2024-01/Assembling-a-Group-of-Products_508c_0.pdf

[17] https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=285454

[18] https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=285411

[19] https://www.cisa.gov/resources-tools/resources/software-transparency-saas-environments-0

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

- ***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 a 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.*
- ***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*
- ***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***: *SBOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external callouts or dynamically loaded components. In some contexts, this may also be referred to as an "Instrumented" or "Dynamic" SBOM*

Additional details on the various types of SBOMs, and the benefits and limitations of each can be found in Appendix B: Summary of Types of SBOMs.

Minimum Elements vs Other Data Elements

The current guidance from the US Department of Commerce identifies three categories of elements: Data Fields, Automation Support, and Practices and Procedures. The current minimum data elements expected to be captured in an SBOM [DOC 2021]:

- ***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.*

As noted in a recent CISA event[21], CISA has been tasked this year to update the minimum elements. As of the writing of this white paper, it is unknown what that task entails or how it affects the current guidance.

The Minimum Elements guidance also provides insight into recommended elements as well. The additional data elements are:[DOC 2021]

- ***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. They also note, "If component information was obtained from a tool that did not have direct access to the underlying component (e.g. a binary analysis tool), then the component author may not be able to credibly determine the exact bits used, and so be unable to generate a hash."*
- ***Lifecycle Phase*** *The data about software components can be collected at different stages in the software lifecycle, including from the software source, at build time, or after build through a binary analysis tool.*
- ***Other Component Relationships:*** *Other types of dependency relationships can be captured and have been implemented in some SBOM standards. One approach that can be captured today beyond direct dependencies is "derivation" or "descendancy". This can indicate that a component is similar to some other known component, but that some changes have been made. It can be useful to track for its shared origins and content.*
- ***License Information***[22]*: SBOMs can convey data about the licenses*[23] *for each component. This data can also allow the user or purchaser to know if the software can be used as a component of another application without creating legal risk.* It should be noted the licenses captured by the SBOM tool are not the same as the data rights as established in the contract.

Each of the previously referenced standards (SPDX, CycloneDX, and SWID) were developed independently prior to the Executive Order and have additional fields that could possibly be of use depending on the questions being asked For example, the CycloneDX format also supports the provenance tracking of software products and their components. This makes it easier to identify the authors and suppliers of software and all its components. [Brudo 2024].

In Appendix A, there is a mapping of the minimum elements to the three designated standards and which specific fields those elements are implemented in. The table also includes the recommended elements and mapping.

[21] *SBOM-a-Rama Winter 2024*, 29 Feb 2024

[22] SPDX V2.2 and beyond support the additional recommended data elements. https://insights.sei.cmu.edu/documents/5319/2023_017_001_978949.pdf

[23] https://spdx.org/licenses/ CycloneDX has adopted the licensing structure and data from SPDX as part of their implementation of the licensing information.

Office of Management and Budget (OMB) Guidance

The Office of Management of Budget (OMB), in response to EO 14028, issued two different memorandums relevant to SBOMs:

- OMB M-22-18, *Enhancing the Security of the Software Supply Chain through Secure Software Development Practices*[24]
- OMB M-23-16, *Update to Memorandum M-22-18, Enhancing the Security of the Software Supply Chain through Secure Software Development Practices*[25]

The first memorandum, OMB M-22-18, *"...requires each Federal agency to comply with the NIST Guidance when using third-party software on the agency's information systems or otherwise affecting the agency's information.*

The term "software" for purposes of this memorandum includes firmware, operating systems, applications, and application services (e.g., cloud-based software), as well as products containing software." [OMB 2022]

The NIST Guidance is referring to the adoption of NIST SP 800-218, NIST Secure Software Development Framework (SSDF).[26] The SSDF has an entry in Table 1 referring to SBOM. It shows the following as part of the **Archive and Protect Each Software Release (PS.3)** practice: [NIST 2022]

> ***TASK****: PS.3.2: Collect, safeguard, maintain, and share provenance data for all components of each software release (e.g., in a software bill of materials [SBOM]).*
>
> ***NOTIONAL IMPLEMENTATION EXAMPLES:***
>
> - ***Example 1****: Make the provenance data available to software acquirers in accordance with the organization's policies, preferably using standards-based formats.*
> - ***Example 2****: Make the provenance data available to the organization's operations and response teams to aid them in mitigating software vulnerabilities.*
> - ***Example 3****: Protect the integrity of provenance data and provide a way for recipients to verify provenance data integrity.*
> - ***Example 4****: Update the provenance data every time any of the software's components are updated*

One of the key activities in the memorandum is the submission of self-attestations from the developers/providers that secure coding practices were followed. It further discusses software artifacts that demonstration conformance to secure software development practices: [NIST 2022]

[24] https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf

[25] https://www.whitehouse.gov/wp-content/uploads/2023/06/M-23-16-Update-to-M-22-18-Enhancing-Software-Security-1.pdf

[26] https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-218.pdf

a. A Software Bill of Materials (SBOMs) may be required by the agency in solicitation requirements, based on the criticality of the software as defined in M-21-30, or as determined by the agency. If required, the SBOM shall be retained by the agency, unless the software producer posts it publicly and provides a link to that posting to the agency.

b. SBOMs must be generated in one of the data formats defined in the National Telecommunications and Information Administration (NTIA) report "The Minimum Elements for a Software Bill of Materials (SBOM)," or successor guidance as published by the Cybersecurity and Infrastructure Security Agency (CISA).

c. Agencies shall consider reciprocity of SBOM and other artifacts from software producers that are maintained by other Federal agencies, based on direct applicability and currency of the artifacts.

d. Artifacts other than the SBOM (e.g., from the use of automated tools and processes which validate the integrity of the source code and check for known or potential vulnerabilities) may be required if the agency determines them necessary.

e. Evidence that the software producer participates in a Vulnerability Disclosure Program may be required by the agency.

f. Agencies are encouraged to notify potential vendors of requirements as early in the acquisition process as feasible, including leveraging pre-solicitation activities.

The second OMB memorandum provided additional guidance and clarifications when from the previous memorandum in three areas: Third-Party Components, Freely Obtained and Publicly Available Proprietary Software, and Federal Contractor Developed Software.[OMB 2023]

- ***Third Party Components*** *- Attestations must be collected from the producer of the software end product used by an agency because the producer of that end product is best positioned to ensure its security. An attestation provided by that producer to an agency serves as an affirmative statement that the producer follows the secure software development minimum requirements, as articulated in the common form.*
- ***Freely Obtained and Publicly Available Proprietary Software*** *- Agencies are not required to collect attestations from software producers for products that are proprietary but freely obtained and publicly available. Open-source software freely and directly obtained by Federal agencies is outside the scope of NIST's guidance for agencies on software supply chain security.*
- ***Federal Contractor Developed Software*** *- Agency-developed software remains out of scope for M-22-18 and any attestation collection requirements. Whether software developed under a Federal contract may constitute "[a]gency-developed software" for the purposes of M-22-18, as amended, depends on whether the contracting agency is able to ensure that secure software development practices are followed throughout the entire software development lifecycle (i.e., requirements, design, development, testing, deployment, and maintenance).*

Agencies, in their development of software, are expected to appropriately leverage the NIST SSDF (SP 800-218).

The applicability of the OMB memorandums may depend on the type of system undergoing operational test and evaluation (OT&E). Typically, OMB memorandums apply to Defense Business Systems but not necessarily National Security Systems with a few exceptions (e.g. financial reporting). In the absence of further definitive guidance, either from the DoD or OMB, it will be up to the organization/agency acquiring the software to make the determination regarding the applicability.

Recent Developments

National Security Administration (NSA)

The NSA recently released a publication, *Recommendations for Software Bill of Materials (SBOM) Management*, which contains specific recommendations applicable to National Security Systems (NSS). These recommendations are based on research and testing of SBOM tools as part of a Cybersecurity Supply Chain Risk Management (C-SCRM) strategy [NSA 2024]. Although the recommendations for NSS are relative to specific contract language, they are still applicable to OT&E activities.

Joint Federated Assurance Center (JFAC)

The Joint Federated Assurance Center (JFAC) sponsored an effort to perform an assessment[27] of 17 different Software Composition Analysis/SBOM tools that were currently on the market and available to DoD customers. Their approach was [JFAC 2024]:

1. *Conduct market research to identify available SBOM/SCA tools*
2. *Identify alignment of tools to software development life cycle*
3. *Implement repeatable benchmark testing and identify public code repositories*
4. *Research available literature to establish a comprehensive evaluation criteria*
5. *Evaluate capabilities and limitations of SBOM/SCA tools*
6. *Evaluate performance of SBOM creation outputs*

The findings from the assessment were:

The SBOM/SCA area is rapidly evolving, many vendors are investing to mature their tools

- *There is no clear winning analysis tool –leverage multiple tools most suited to your needs*
- *SBOM data provided by tools is not consistent*

[27] https://jfac.apps.dso.mil/assessments/details/1535

- *Only one tool (FOSSA) provided the data for the minimum element fields in SPDX format for all components*
- *Not many tools offer support for SBOM attestation capabilities*
- *The ability to parse binaries with no package manager is still a huge challenge for most tools*
- *SBOM/SCA tools working in multi-language repositories warrants more study*

NNSA/DoD Software Assurance Community of Practice

There is an SBOM Working Group conducting efforts within the DoD and the Nuclear National Security Administration (NNSA) within Department of Energy (DoE) to develop an SBOM Technical Guidance & Recommendations document[28]. The SBOM Working Group has produced a draft working copy that also contains policy recommendations relative to SBOMs for the following DoD Instructions: [NNSA/DoD SWA CoP 2024]

- *DoDI 5000.83, Technology and Program Protection Plan*
- *DoDI 5200.44, Protection of Mission Critical Functions to Achieve Trusted Systems and Networks*
- *DoDI 5200.47, Anti-Tamper*
- *DoDI 5000.89, Test and Evaluation*
- *DoDI 5000.90, Cybersecurity for Acquisition Decision Authority and Program Managers*
- *DoDI 8500.01, Cybersecurity*

Vulnerability Exploitation eXchange (VEX)

There is an SBOM related concept called Vulnerability Exploitation eXchange (VEX) that is starting to gain acceptance in the commercial sector. A VEX document is an attestation, a form of security advisory that indicates whether a product or products are affected by known vulnerability or vulnerabilities.[29] As noted by CISA,

> *While the VEX concept was developed to fill a particular need regarding use of software bills of materials (SBOMs), VEX is not limited to use with SBOMs or necessarily expected to be included in the SBOM itself.*
>
> *The primary use cases for VEX are to provide users (e.g., operators, developers, and services providers) additional information on whether a product is impacted by a specific vulnerability in an*

[28] https://intelshare.intelink.gov/sites/dodswawg/_layouts/15/WopiFrame2.aspx?sourcedoc=%7b2B2C5743-9315-4C6D-9138-F4704B161FCD%7d&file=Technical%20Guidance%20%26%20Recommendations%202%20Draft%201.docx&action=default

[29] https://www.cisa.gov/sbom

included component and, if affected, whether there are actions recommended to remediate. In many cases , a vulnerability in an upstream component will not be "exploitable" in the final product for various reasons (e.g., the affected code is not loaded by the compiler, or some inline protections exist elsewhere in the software). [CISA 2021]

VEX standards are also a work in progress and has their minimum compliance elements. Further information on VEX is available from CISA.[30]

Relationship between SBOM Tools and Software Composition Analysis (SCA) Tools

While SBOM tools and SCA tools were initially distinctly different, there is a growing overlap in the abilities of those tools. SBOM tools provide a comprehensive list of all the components and dependencies that constitute a particular software application. SCA tools analyze the software's codebase, detect third-party dependencies, and provide insights into the security posture of those components. [Paliwal 2023] According to Paliwal, there are some differences in the tools. Those differences are shown in *Table 2*.

Table 2: Differences between SBOM and SCA Tools (adapted)

	SBOM Tools	**SCA Tools**
Scope	Provides a comprehensive list of all components used in a software application, including open-source and proprietary dependencies.	Specifically focused on analyzing and managing open-source components and their associated vulnerabilities.
Functionality	Serves as a complete inventory of software components, allowing for supply chain management, risk assessment, and compliance.	Primary function is to identify, track, and address security vulnerabilities and licensing issues associated with open-source components. Often provide continuous monitoring capabilities
Use Cases	Useful for various stakeholders, including developers, security teams, and users, as it offers a holistic view of software composition.	Primarily targeted at developers and security teams, helping them ensure the security and compliance of open-source components.

There has been a noticeable movement recently in the commercial world to add functions typically found in an SCA tool into an SBOM tool.

SBOM & HBOM Efforts

Before the advent of SBOMs, hardware developers would include a Bill of Materials (BOM) detailing the parts and part composition of the hardware. The DoD has a Data Item Description (DID) for BOM dating back to 2005 (DI-PSSS-8165B, *Bill of Materials (BOM) for Logistics and Supply Chain Risk*

[30] *Minimum Requirements for Vulnerability Exploitability eXchange (VEX),* https://www.cisa.gov/sites/default/files/2023-04/minimum-requirements-for-vex-508c.pdf

Management).[31] This DID contains some similar information to that of an SBOM (e.g., OEM Name, Software/Firmware Version Number) but most of the information is hardware specific. Not unexpected, it also has information regarding security clearance levels of the OEM Facility as well as that of the Manufacturer of the component.

There has been a new effort from CISA for the development of a *Hardware Bill of Materials (HBOM) Framework for Supply Chain Risk Management*.[32] . According to CISA, The HBOM Framework provides basic information about including the firmware associated with the products' components (i.e., the provider of the firmware), but stops short of proposing a framework for examining the provenance and other attributes of that firmware. [CISA 2023b] The CISA HBOM effort notes the SBOM information is out of scope, but rather endeavors "to ensure consistency with other frameworks that are becoming prominent approaches to providing SBOM, such as CycloneDX and SPDX." The CISA HBOM document contains an appendix (Appendix C – HBOM Taxonomy) that attempts to map the HBOM field names to equivalent fields in the SDPX and CycloneDX standards. The CISA effort in this area does differ significantly (document contents) than the BOM DID.

It should also be noted there is an effort from the DoD, sponsored by the offices of the Chief Information Officer (CIO) and Chief Data and Artificial Intelligence Office (CDAO) to create a multipurpose Extensible Bill of Materials (xBOM) that would address HBOMs and SBOMs in a single standard.[33][34]

SBOM only vs. SBOM Combined with Additional Information

As depicted earlier in Figure 1, an SBOM itself contains useful information, however when combined with additional information not contained in the SBOM, it reveals or contributes to an additional awareness or understanding. Currently there is a lot of focus on using SBOMs combined with vulnerability data. The combination of those two sets of data will yield an initial awareness if a vulnerability is contained within the system. However, the identification of a vulnerability still requires additional information or knowledge such as where is the vulnerable component is located within the system & software architecture. This is part of the analysis challenge beyond just obtaining the SBOMs.

[31] https://quicksearch.dla.mil/Transient/3B223C30507D44B1A56CC234C7C08B0B.pdf

[32] https://www.cisa.gov/sites/default/files/2023-09/A%20Hardware%20Bill%20of%20Materials%20Framework%20for%20Supply%20Chain%20Risk%20Management%20%28508%29.pdf

[33] https://www.youtube.com/watch?v=IRoifXWT5BU

[34] DoD Memorandum, *Extensible Bills of Materials Strategy for the Digital Management of Department of Defense Assets* (draft)

General SBOM Challenges and Lessons Learned

The adoption and implementation of SBOMs is still very much in its infancy and not without challenges. Some of those challenges were encountered and documented by the Southern Company in Mississippi.[35] They decided to inventory all of the hardware, software and firmware in equipment running it one of its Mississippi substations in order to create a Software Bill of Materials for the operational technology. Some of the poignant lessons learned were [Higgins 2024]:

- *Sixty percent of the vendors declined to provide the information*
- *It took an average of 60 days and dozens of meetings to receive SBOM from cooperative vendors*
- *Trust but verify the SBOM data (missing component and dependency data)*
- *Use SBOMs with vulnerability databases to vet the information and vulnerabilities as well as the potential exploitability*
- *Some contract restructuring will possibly be needed to include SBOM requirements*
- *Additional analysis is needed beyond the SBOM data. "If you're just collecting SBOMs and can't do anything with the data, they are just JSON documents in a folder."*

General Questions to Ask

Some general questions to think about when using SBOMs for OT&E activities:

Are there SBOM(s) available for the system under test? One or more SBOMs needed may not necessarily be available. Instead, there may be an attestation/affirmation that the "upstream" information is not available. If the system is a legacy system, the SBOM tools might not support older languages, driving the SBOM generation to a manual function.

When was the SBOM generated? An SBOM could be manually created, or tool generated. Discovering when in the development process was created is useful, particularly with respect to the currency of the information. This directly relates to the types of SBOMs (as noted earlier). An SBOM from source may not necessarily show all the dependencies of an SBOM generated at build time. An SBOM from the build may not necessarily show the runtime dependencies for the operational environment. From an operational environment perspective, it may be necessary to generate an SBOM from the binary and validate it against the implementation.

Does the SBOM show a decomposition of the third-party components below the primary (initial) level (e.g., secondary, and tertiary dependencies)? The first level of dependencies should be very easy for the software developer to know (e.g., what components they used in the delivered system).

[35] https://www.darkreading.com/ics-ot-security/southern-company-builds-a-power-substation-sbom

What is included in those components may not necessarily be known and the information not actually available.

Is the SBOM complete (no missing information) and correct (e.g., version numbers, names of components, supplier information, etc.)? The Program Office would be expected to validate this information. Is the expected information complete, correct, and understandable? Is the version information in an understandable format? The version number, typically of the format *<major>.<minor>.<patch>.<build>* should be expressed. [Hissam 2024] Alternatives may include:

- *commit IDs (e.g., 0.0.0-20181124034731-591f970eefbb, 57.0.0+incompatible) These may be more problematic when attempting to derive additional (upstream/transitive) dependencies or enumerated vulnerabilities based on that specific (commit ID) in common vulnerability databases.*
- *semantic versioning (e.g., >= 0.12 < 0.13, 1.x || ~0.12.1): These may be more problematic when attempting to derive additional (upstream/transitive) dependencies or enumerated vulnerabilities based on that specific (semantic version) in common vulnerability database*

Can the traceability from the components to Common Vulnerability Enumeration (CVEs)[36] be done easily? The unique component naming conventions to allow for tracing are not necessarily consistent. There is the potential for more than one naming convention to be provided Examples of commonly used unique identifiers are Package Uniform Resource Locators (PURL)[37].Common Platform Enumeration (CPE)[38], and Software Identification (SWID)[39] tags. [DOC 2021]

- ***A PURL or package URL** is an attempt to standardize existing approaches to reliably identify and locate software packages. A PURL is a URL string used to identify and locate a software package in a mostly universal and uniform way across programing languages, package managers, packaging conventions, tools, APIs and databases. Such a package URL is useful to reliably reference the same software package using a simple and expressive syntax and conventions based on familiar URLs* [GitHub 2024]. *For example, pkg:npm/@ampproject/remapping@2.2.0) has the context free advantage of explicit conveying the language/package ecosystem of the component (npm[40] in the example shown here) which is effective, at least from the standpoint of identifying the upstream "source" (e.g., an open-source repository) for the component is easier as*

---

[36] https://www.cve.org/

[37] See *Software Identification Challenges and Guidance, supra note 9*; Package-url/purl-spec, GitHub, https://github.com/package-url/purl-spec [DOC 2021]

[38] See Framing Working Group, Nat'l Telecomms. & Info. Admin., *Software Identification Challenges and Guidance* (2021), *https://www.ntia.gov/files/ntia/publications/ntia_sbom_software_identity-2021mar30.pdf Official Common Platform Enumeration (CPE) Dictionary*, Nat'l Inst. Standards & Tech., https://nvd.nist.gov/products/cpe [DOC 2021]

[39] See *Software Identification Challenges and Guidance, supra note 9; ISO/IET 19770-2:2015 Information Technology–IT Asset Management—Part 2: Software Identification Tag*, Int'l Standards Org., https://www.iso.org/standard/65666.html [DOC 2021]

[40] https://docs.npmjs.com/about-npm

opposed to the other unique identifiers which would satisfy this requirement [Hissam 2024]. *Identification at the package level may not necessarily be needed from an OT&E perspective.*

- ***A CPE** is a structured naming scheme for information technology systems, software, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name.[NIST 2024a] For example, cpe:2.3:a:\\@ampproject\\/remapping:\\@ampproject\\/remapping:2.2.0:*:*:*:*:*:*:*. A CPE allows a convenient mapping to entries in the National Vulnerability Database (NVD) and Common Vulnerability Enumeration (CVEs).* The downside to the CPE is the lack some of the details which are included in a PURL. [Hissam 2024].
- ***A SWID tag** document is composed of a structured set of data elements that identify the software product, characterize the product's version, the organizations and individuals that had a role in the production and distribution of the product, information about the artifacts that comprise a software product, relationships between software products, and other descriptive metadata. The information in a SWID tag provides software asset management and security tools with valuable information needed to automate the management of a software install across the software's deployment lifecycle[41]. SWID tags support automation of software inventory as part of a software asset management (SAM) process, assessment of software vulnerabilities present on a computing device, detection of missing patches, targeting of configuration checklist assessments, software integrity checking, installation and execution whitelists/blacklists, and other security and operational use cases.* [NIST 2024b]. It should be noted that SWID tags are also referred to as one of the three named standards for consideration for SBOMs in general as well as for unique identifiers within an SBOM. There is a NIST effort underway to include SWID tags for vulnerability datasets provided by the NVD allowing another method to cross-reference products similar to CPEs.[42]

What standard does the SBOM(s) use? There are currently three different referenced standards for SBOMs that have different representations of the data. In the case of multiple SBOMs, can the tool(s) used ingest and use multiple formats? If not, can the SBOM data be converted possibly through the use of a third-party tool?

Have the SBOM(s) been validated against the minimum elements for compliance? As noted earlier, compliance with the minimum data elements requirement is an issue. Of the 17 products evaluated in the JFAC assessment, only 1 fully complied with the minimum elements. The others either had missing/incomplete information or didn't support the minimum elements. There is at least one open-source tool (NTIA Conformance Checker)[43] that validates SBOM compliance with the minimum elements but it only addresses the SPDX standard.

[41] https://csrc.nist.gov/projects/software-identification-swid/lifecycle

[42] https://csrc.nist.gov/Projects/Software-Identification-SWID

[43] https://github.com/spdx/ntia-conformance-checker

Does the SBOM contain vulnerability information as part of the SBOM data? If so, it is most likely out of date almost immediately given how quickly vulnerabilities are discovered and reported. Ideally the vulnerability information should be published separately from the SBOM itself possibly through a VEX document as noted earlier.

Proposed SBOM Use Cases of Operation Test and Evaluation Activities

Based on various background documents identified earlier in this white paper most of the use cases that pertain to Operational Test and Evaluation fall into one or more of the following areas:

- Software/System configuration including any runtime dependencies
- Validation/confirmation of the delivered software/system configuration
- Known vulnerabilities
- Foreign Ownership, Control, or Influence (FOCI)

Operational test personnel don't typically have access to the source code but some identification of the software information (components, versions, system/software architecture, etc.) is still needed as part of the vulnerability analysis based on the information in the Cybersecurity Test & Evaluation Guidebook.

The third-party component dependencies are of particular concern especially if those components comprise part of a critical function (partially or wholly) within the system under test. This concern is also reflected in the following Common Weakness Enumerations (CWEs)[44]:

- CWE-1357: *Reliance on Insufficiently Trustworthy Component*: https://cwe.mitre.org/data/definitions/1357.html
- CWE-1104: *Use of Unmaintained Third Party Components*: https://cwe.mitre.org/data/definitions/1104.html
- CWE-1329: *Reliance on Component That is Not Updateable*: https://cwe.mitre.org/data/definitions/1329.html
- CWE-1395: *Dependency on Vulnerable Third-Party Component*: https://cwe.mitre.org/data/definitions/1395.html

Use Case 1: Build/Configuration Identification (Internal Dependencies)

Description: As an operational tester, I want to validate the internal dependencies in the software under test so I can confirm the correct components have been properly identified.

Rationale: Understanding what is included internally in the delivered software is key. This can be used to determine if there is a mismatch between the delivered software (e.g., one or more specific

[44] Common Weakness Enumeration (CWE™) is a community-developed list of common software and hardware weaknesses. A "weakness" is a condition in a software, firmware, hardware, or service component that, under certain circumstances, could contribute to the introduction of vulnerabilities. The CWE List and associated classification taxonomy identify and describe weaknesses in terms of CWEs.[MITRE 2024]

components) and the SBOM (doesn't show the components or the version information is not correct) yielding an incorrect confirmation.

Challenges:

- Depending on when in the development lifecycle the SBOM was generated and type of SBOM not all the component information may be included (e.g., runtime dependencies).
- There still may be a need to generate an SBOM from a binary and do a comparison to confirm the provided SBOM matches the implementation. Generation of an SBOM could be a challenge for air gapped systems as very few SBOM tools currently available provide this ability.
- The level of decomposition of the dependencies (e.g., secondary, and tertiary dependencies) may not necessarily be known.

Use Case 2: Runtime Configuration Identification (External Dependencies)

Description: As an operational tester, I want to validate the SBOM contains the external (runtime) dependencies to confirm the target environment is correct

Rationale: Depending on when the SBOM was generated in the development lifecycle, it is entirely possible the external (runtime) dependencies are not necessarily captured in the provided SBOM. Understanding those dependencies in the target test environment is necessary to ensure it is an operationally representational environment.

Challenges:

- Depending on when in the development lifecycle the SBOM was generated and type of SBOM not all the component information may be included (e.g., runtime dependencies).
- There still may be a need to generate an SBOM from a binary and do a comparison to confirm the provided SBOM matches the implementation. Generation of an SBOM could be a challenge for air gapped systems as very few SBOM tools currently available provide this ability.
- Establishing the configuration of the system under test will most likely depend on additional information (e.g., configuration management, etc.) and will require some augmentation of the SBOM data.

Use Case 3: Known Vulnerabilities of Third-Party Components

Description: As an operational tester, I want to identify the various third-party components in the delivered system to analyze it for known vulnerabilities.

Rationale: Knowing what vulnerabilities exist or may exist in the system under test will be necessary as part of the OT&E testing activities (e.g., where, and how is the system vulnerable).

Challenges:

- Needs additional analysis using vulnerability databases[45]
- Knowledge of where in the system and software architecture the components exist
- The identifying information needed for a additional analysis may be unclear, incomplete, or missing from the SBOMs
- If the vulnerability information was included as part of the SBOM data, it will almost certainly be out of date the moment it was published

This is an instance where a separate VEX documents (if they exist) would be useful to augment the vulnerability determination.

Use Case 4: Foreign Ownership, Control, or Influence (FOCI)

Description: As an operational tester, I want to understand if any of the components in the system presents a risk due to foreign ownership, control, or influence.

Rationale: Per the Defense Counterintelligence and Security Agency (DCSA), *A company is considered to be operating under FOCI whenever a foreign interest has the power, direct or indirect, whether or not exercised, and whether or not exercisable, to direct or decide matters affecting the management or operations of that company in a manner which may result in unauthorized access to classified information or may adversely affect the performance of classified contracts*[46].

This concern is not only for hardware but for software and software components as well that are in the system.

Challenges:

- Not part of the SBOM data elements (or any of the recommended standards). This is primarily a DoD concern so it is not expected that any of the current commercial standards would contain a field pertaining to this concern.
- Requires additional tracing and supply chain intelligence
- Assumes the Supplier information, along with the other relevant elements are complete and correct. There are indicators, "*that may be relevant in identifying FOCI concerns can be derived from several fields, including author, publisher, manufacturer, and supplier but can also be extended to other fields such as the components group name. The CPE may also indicate the vendor and the PURL can identify a potentially foreign namespace or repository or download URL for the package. Many external references may also provide a clue, especially those pointing to*

[45] Based on efforts by MIT/LL Group 52, the inconsistent data standards in the SBOMs will make this analysis much more difficult.

[46] https://www.dcsa.mil/Industrial-Security/Entity-Vetting-Facility-Clearances-FOCI/

the version control system (vcs) and commit history, issue tracker, distribution, and documentation websites."[OWASP 2023]

Use Case 5: Validation of Dependencies (Binary SBOM Generation)

Description: As an operational tester, I want to validate the SBOM information that was provided against the system that was delivered for testing

Rationale: Depending on how and when the SBOM was generated, it could potentially contain information that is out of date (e.g., incorrect version number, etc.). In order to confirm SBOM matches the delivery an SBOM may need to be generated from the binary files and then a comparison with the SBOMs that were delivered.

Challenges:

- Depending on when the delivered SBOMs were built, they may differ from the results of a binary analysis and generation.
- *Compilers often strip properties important for SBOM analysis out of the binary. The most common example is version information. While version information is present some of the time, quite often it is missing which means assessing is a vulnerability is present again requires source code.* [Hoog 2022]
- *While binary analysis is often able to detect the presence of components, it becomes extremely difficult to build deeply nested dependencies trees. So, we might be able to determine openssl is present but which code included it is not always available*.[Hoog 2022]
- Many of the SBOM tools will not support binary analysis to generate SBOMs. In the JFAC tool assessment, only 4 of the 17 tools supported binary analysis, and 3 had limited support.
- There may be issues with air gapped environments that would limit the SBOM tool choice in some instances.

Conclusions and Recommendations

Conclusions

- The creation and use of SBOM is still very much in an early adoption phase, particularly within industry. The DoD is slightly further behind industry at the moment (just beginning to think about SBOMs in light of EO 14028). There is still some significant industry pushback on creating and releasing SBOMs.
- The SBOM quality is still an issue with many of the tools (missing or incomplete data).
- The capabilities of SCA tools and SBOM tools are starting to overlap in many areas.
- SBOMs on their own provide only a limited set of information and will require additional analysis and data.

The JFAC tool assessment yielded the following conclusions [JFAC 2024]:

- *The SBOM/SCA area is rapidly evolving, many vendors are investing to mature their tools*
- *There is no clear winning analysis tool –leverage multiple tools most suited to your needs*
- *SBOM data provided by tools is not consistent*
- *Only one tool (FOSSA) provided the data for the minimum element fields in SPDX format for all components*
- *Not many tools offer support for SBOM attestation capabilities*
- *The ability to parse binaries with no package manager is still a huge challenge for most tools*
- *SBOM/SCA tools working in multi-language repositories warrants more study*

Recommendations

1. More than a single SBOM tool should be considered. Use the current JFAC tool assessment to compare the capabilities of the tools to make an informed decision. As noted earlier, very few of the tools can work in an air-gapped environment should that be a testing consideration.
2. Be prepared to perform the additional analysis based on the SBOM data. Identifying vulnerability information from the component name will require additional analysis & research. Looking for cross-component dependencies would be an initial start. If the SBOM tool(s) used by the operational testers can graph the dependencies, it would provide a significant improvement over text/table-based data in locating that information.
3. Validate the SBOM contents for conformance with the minimum elements requirement.
4. Determine what type of SBOM(s) are being provided. An SBOM that was tool created at Build time utilizing the source code (e.g., a Build SBOM) should be the minimum acceptable version. An Analyzed or Runtime SBOM would be even better, however it is more likely that the operational testers will have to create that type of SBOM by performing the additional analysis to augment the SBOM(s).

5. Determine if the SBOM has been digitally signed or authenticated hashes of the components are available.
6. Determine if any agreements are needed to perform a binary analysis of the software. A binary analysis may fall into the legal area of “reverse engineering”.
7. Determine if any VEX documents exist for components included in the system under test.

Appendix A: Mapping of SBOM Minimum Elements to Appliable Standards

Table 3: Mapping of the SBOM Minimum Elements to Referenced Standards[47]

Attribute	Description	Mandatory or Recommended	SPDX (v2.3) Field	CycloneDX Field	SWID Field
Author Name	The name of the entity that create the SBOM data for this component.	M	`Creator`	`metadata/authors/author`	`<Entity> @role (tagCreator), @name`
Timestamp	Record of the date and time of the SBOM data assembly	M	`Created`	`metadata/timestamp`	`<Meta>`
Supplier Name	The name of an entity that create, defines, and identified the components	M	`PackageSupplier`	`Supplier` `publisher`	`<Entity> @role (softwareCreator/publisher), @name`
Component Name	Designation assigned to a unit of software defined by the original supplier	M	`PackageName`	`name`	`<softwareIdentity> @version`
Version String	Identifier used by the supplier to specify a change in the software from a previously identified version	M	`PackageVersion`	`version`	`<softwareIdentity> @version`
Component Hash	A cryptographic hash	R	`PackageChecksum or VerificationCode`	`bom.components[].hashes[]`	`<Payload>/../<file> @ [hash-algorithm]:hash`

[47] Derived from: https://www.ntia.gov/files/ntia/publications/sbom_formats_survey-version-2021.pdf

Attribute	Description	Mandatory or Recommended	SPDX (v2.3) Field	CycloneDX Field	SWID Field
Unique Identifier	Other identifiers that are used to identify a component, or serve as a lookup key for relevant databases	M	`DocumentNamespace combined with SPDXID`	`bom/serialNumber component/bom-ref`	`<softwareIdentity> @tagID`
Relationship	Characterizing the relationship that an upstream component X is included in Y	M	`Relationship: DESCRIBES; CONTAINS`	`(Inherent in nested assembly/subassembly and/or dependency graphs)`	`<Link> @rel, @href`
Lifecycle Phase	Conveys where, when, and how the SBOM data was re-corder. Simply noting how the data was captured (e.g., "source", "build", or "post-build") will be helpful for consumption and data management	R	`LifecycleScopeType`	`bom.metadata.lifecycles[]`	(Maintained over the lifecycle phase with primary, supplemental, and patch information)
License Information	The license and terms for each component.	R	`LicenseRef-[idstring]`	`bom.components[].licenses[]`	`<softwareIdentity> <link rel="license" href=url>`

Appendix B: Summary of Types of SBOMs

Table 4: Types of SBOMs, Benefits and Limitations of Each Type[48]

SBOM Type	Definition	Data Description	Benefits	Limitations
Design	**SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.**	Typically derived from a design specification, RFP, or initial concept	• Highlight incompatible components ahead of licensing purchase or acquisition. • Defines approved or recommended included component list for developer use	• This may be very difficult to generate. • Unlikely to identify as much detail as found in other SBOM types
Source	**SBOM created directly from the development environment, source files, and included dependencies used to build a product artifact**.	Typically generated from software composition analysis (SCA) tooling, with manual clarifications.	• Provides visibility without access to build process. • Can facilitate remediation of vulnerabilities at the source. • Can provide a view into the dependency tree / hierarchy of the included components	• Can highlight components (which might have vulnerabilities) that never run or are compiled out in deployed code. • Depending on language/ecosystem, may not include runtime, plugin, or dynamic components, like appserver or platform libraries. • May require references to other SBOMs for completeness.
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.**	Typically generated as part of a build process. May consist of integrated intermediate Build and Source SBOMs for a final release artifact SBOM.	• Increases confidence that the SBOM representation of the product artifact is correct due to information available during the build and/or Continuous Integration/Continuous Deployment (CI/CD) processes. • Provides visibility into more components than just source code. • Increased trust by enabling signing of the SBOM and product artifact by the same build workflow.	• Potentially have to change the build process to generate this SBOM. • Highly dependent on the build environment in which the build is executed. • May be difficult to capture indirect and/or runtime dependencies. • May not contain the correct versions of dynamically linked dependencies (as they may be replaced at runtime depending on language/ecosystem).

[48] Adapted from: https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

SBOM Type	Definition	Data Description	Benefits	Limitations
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	Typically generated through analysis of artifacts by 3rd party tooling.	• Provides visibility without an active development environment, such as legacy firmware artifacts. • Does not need access to the build process. • Can help verify SBOM data from other sources. • May find hidden dependencies missed by other SBOM type creation tools.	• May be prone to omissions, errors, or approximations if the tool is unable to decompose or recognize the software components precisely. • May depend on heuristics or context-specific risk factors.
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.**	Typically generated by recording the SBOMs and configuration information of artifacts that have been installed on systems.	• Highlights software components installed on a system, including other configurations and system components used to run an application	• May require changing install and deploy processes to generate. • May not accurately reflect the software's runtime environment, as components may reside in inaccessible code.
Runtime	**SBOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external callouts or dynamically loaded components**. In some contexts, this may also be referred to as an "Instrumented" or "Dynamic" SBOM	Typically generated from tooling interacting with a system to record the artifacts present in a running environment and/or that have been executed.	• Provides visibility to understand what is in use when the system is running, including dynamically loaded components and external connections. • Can include detailed information about whether components are active and what parts are used.	• Requires the system to be analyzed while running, which may require additional overhead. • Some detailed information may be available only after the system has run for a period of time until the complete functionality has been exercised.

References

URLs are valid as of the publication date of this report.

[Brudo 2024]
Brudo, Barak, *SPDX vs. CycloneDX: SBOM Formats Compared*, Scribe, https://scribesecurity.com/blog/spdx-vs-cyclonedx-sbom-formats-compared/

[CISA 2021]
Cybersecurity & Infrastructure Security Agency (CISA), *Vulnerability-Exploitability eXchange (VEX) – An Overview*, 27 September 2021, Department of Homeland Security, https://www.ntia.gov/files/ntia/publications/vex_one-page_summary.pdf

[CISA 2023a]
Cybersecurity & Infrastructure Security Agency (CISA), *Types of Software Bill of Material (SBOM) Documents*, Department of Homeland Security, April 2023, https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf

[CISA 2023b]
Cybersecurity& Infrastructure Security Agency (CISA), *A Hardware Bill of Materials (HBOM) Framework for Supply Chain Risk Management*, Department of Homeland Security, September 2023, https://www.cisa.gov/resources-tools/resources/hardware-bill-materials-hbom-framework-supply-chain-risk-management

[DOC 2021]
United States Department of Commerce (DOC), *The Minimum Elements For a Software Bill of Materials (SBOM)*, 2021.
https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

[EO 14025]
Executive Order 14025, *Executive Order on Improving the Nation's Cybersecurity,* , The White House, 12 May 2021, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[GitHub 2024]
Software Identification Challenges and Guidance, GitHub, https://github.com/package-url/purl-spec

[Higgins 2024]
Kiggins, Kelly Jackson, *Southern Company Builds SBOM for Electric Power Substation*, 6 March 2024, Dark Reading, https://www.darkreading.com/ics-ot-security/southern-company-builds-a-power-substation-sbom

[Hissam 2024]
Hissam, Scott, *SBOM State of the Practice*, Software Engineering Institute (SEI)

[Hoog, 2022]
Hoog, Andrew, *Source Code vs. Binary Analysis for SBOMs*, 5 Oct 2022, https://www.andrewhoog.com/post/source-code-vs-binary-analysis-for-sbom/

[JFAC 2024]
Joint Federated Assurance Center (JFAC), *FY23/24 Software Composition Analysis/SBOM Tool Assessments*, 2024, https://jfac.apps.dso.mil/assessments/details/1535

[MITRE 2024]
MITRE Corporation, *Common Weakness Enumeration: About CWE*, https://cwe.mitre.org/about/index.html

[NIST 2021]
National Institute of Standards and Technology (NIST), *Survey of Existing SBOM Formats and Standards*, , United States Department of Commerce, 2021, https://www.ntia.gov/files/ntia/publications/sbom_formats_survey-version-2021.pdf

[NIST 2022]
National Institute of Standards and Technology (NIST), *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*, NIST SP 800-218, https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-218.pdf

[NIST 2024a]
National Institute of Standards and Technology (NIST), *National Vulnerability Database (NVD) Official Common Platform Enumeration (CPE) Dictionary*, https://nvd.nist.gov/products/cpe

[NIST 2024b]
National Institute of Standards and Technology (NIST), *National Vulnerability Database (NVD) Software Identification Tags (SWID Tags)*, https://nvd.nist.gov/products/swid

[NNSA/DoD SWA CoP 2024]
NNSA/DoD Software Assurance Community of Practice, SBOM Working Group, *SBOM Technical Guidance & Recommendations(draft)*, 2024, https://intelshare.intelink.gov/sites/dodswawg/_layouts/15/WopiFrame2.aspx?sourcedoc=%7b2B2C5743-9315-4C6D-9138-F4704B161FCD%7d&file=Technical%20Guidance%20%26%20Recommendations%202%20Draft%201.docx&action=default

[NSA 2024]
National Security Administration (NSA), *Recommendations for Software Bill of Materials (SBOM) Management,* Version 1.1, January 2024, https://media.defense.gov/2023/Dec/14/2003359097/-1/-1/0/CSI-SCRM-SBOM-Management-v1.1.PDF

[OMB 2022]
Office of Management and Budget (OMB), *Enhancing the Security of the Software Supply Chain through Secure Software Development Practices, OMB M-22-18,* 14 Sep 2022, https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf

[OMB 2023]
Office of Management and Budget (OMB), *Update to Memorandum M-22-18, Enhancing the Security of the Software Supply Chain through Secure Software Development Practices*, *OMB M-23-16*, 9 June 2023, https://www.whitehouse.gov/wp-content/uploads/2023/06/M-23-16-Update-to-M-22-18-Enhancing-Software-Security-1.pdf

[OWASP 2023]
Open Worldwide Application Security Project (OWASP), *Authoritative Guide to SBOM: Implement and optimize use of Software Bill of Materials*, 25 June 2023, https://cyclonedx.org/guides/OWASP_CycloneDX-Authoritative-Guide-to-SBOM-en.pdf

[Paliwal 2023]
Paliwal, Ashwani, *SBOM vs. SCA*, SecOps Solution, 7 Sep 2023, https://www.secopsolution.com/blog/sbom-vs-sca

[Wallen 2023]
Wallen, Charles; Alberts, Christopher; Bandor, Michael; Woody, Carol; *Software Bill of Materials (SBOM) Framework: Leveraging SBOMs for Risk Reduction*, Software Engineering Institute, 2023. https://insights.sei.cmu.edu/library/software-bill-of-materials-framework-leveraging-sboms-for-risk-reduction/

Legal Markings

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu
Email: info@sei.cmu.edu