White Paper: Enhancing Software Bill of Materials (SBOM) Generation

This is a draft version of the “Enhancing Software Bill of Materials (SBOM) Generation” White Paper and is in final review by the community.

This document was drafted in an open process by a community of Software Bill of Materials (SBOM) experts, facilitated by the Cybersecurity and Infrastructure Security Agency (CISA). CISA did not draft and is not the author of this document, nor does this document represent an official CISA and/or U.S. Government policy. CISA and the U.S. Government do not specifically adopt or endorse the views expressed in this document.[1]

Abstract

This white paper examines the practical challenges of producing robust, National Telecommunications and Information Administration (NTIA) Minimum Elements-adherent, Software Bills of Materials (SBOM) that not only meet the NTIA Minimum Elements but can go beyond this to meet future compliance frameworks. As of publication, our research found that no single open source tool[2] can reliably generate an SBOM that adheres to NTIA Minimum Elements out of the box**.** We propose a six-step process that separates SBOM creation (or “authoring[3]”) into distinct, manageable phases:

1. **Generation:** Use automated tooling to produce an initial SBOM. This step captures as much component data as possible based on available data.
2. **Augmentation:** Supplement the generated SBOM with critical metadata, such as vendor details or specific environment attributes like main product version information, that the automated tool could not discern on its own.
3. **Enrichment:** Enhance the SBOM by incorporating additional data from external sources, such as open databases/datasets, to fill in missing details and improve the accuracy of dependencies.
4. **Verification**: Ensure the SBOM meets schema and verification requirements prior to signing and releasing the document.
5. **Signing (Optional):** Attest to the SBOM’s integrity by cryptographically signing it, thus ensuring trust and verifiability throughout the software supply chain.

[1] See SBOM Community Legal Explanation
[2] See the section “Tool selection criteria”
[3] See SBOM Sharing Roles and Considerations

6. **Document Linking (Optional):** It's recommended to structure them hierarchically rather than merge them into a single file, grouping components like microservices into their own SBOMs. This results in smaller SBOM implementations that are modular.

By segmenting SBOM creation into these phases, organizations can more effectively identify gaps, integrate multiple data sources, and customize the final artifact to meet evolving regulatory and security requirements. This multi-phase approach allows for flexibility in tool selection, enabling organizations to “swap out” one tool for another without disrupting the end-to-end workflow, while ensuring a higher level of quality, completeness, and trustworthiness in the SBOM.

The SBOM Generation Implementation Tiger Team aimed to deliver a fully functional implementation of the phases outlined above, integrated into both GitHub and GitLab CI/CD pipelines. These implementations include fully operational examples across multiple programming languages, with outputs generated in both CycloneDX and SPDX formats.

Introduction

Software Bill of Materials (SBOM) is a foundational artifact for managing software component transparency and security. It provides a detailed inventory of all components within a software product. With increasing focus on cybersecurity and supply chain risk management, producing high-quality SBOMs is critical. Discussions around data quality, “completeness,” and other SBOM-related issues, coupled with the wide variety of SBOM generation tooling options available in the current marketplace, may dissuade organizations new to SBOM from adopting SBOM generation into their existing software development workflows. Often, teams are unaware of simple additional steps that will elevate the completeness of their SBOMs. This effort aimed to develop SBOM generation workflows that could be replicated or used as a reference for organizations that are new to generating SBOMs or are interested in improving their existing SBOM generation processes. The SBOM Generation References Implementation GitHub repository contains example workflows developed through this effort. This white paper documents the key decisions made in developing these workflows and offers actionable recommendations to address observed challenges for generating “good” SBOMs.

At each stage of the software supply chain, Producers generate SBOMs through the Authoring process and Consumers analyze SBOMs based on their requirements. While much of the interchange of SBOMs is beyond this white paper, it’s important to understand that the relationship between Producer and Consumer creates a series of responsibilities and expectations defining the value received from the SBOM and that SBOM verification is a key part of a Producer’s responsibility. Generated SBOMs follow an SBOM format and are augmented and enriched by information based on additional sources to meet the expectations expressed by a Consumer, such as the NTIA Minimum Elements.

Achieving full compliance may be challenging and individuals should strive for the "best effort" principle to approximate completeness. This white paper intends to outline considerations for

entities authoring SBOMs, with the particular goal of encouraging organizations to begin at least generating SBOMs. As organizations mature in their implementation of SBOM, they may consider the Augmentation, Enrichment, Verification, and additional optional practices to improve the completeness of the SBOM data they author. The SBOM Producer Lifecycle provides a structured framework for creating, enhancing, and sharing Software Bill of Materials (SBOMs) effectively[4]. It is format-agnostic and aligns with guidance from the National Telecommunications and Information Administration (NTIA) and the Cybersecurity and Infrastructure Security Agency (CISA). The lifecycle includes stages such as SBOM generation, where components and dependencies are identified; augmentation and enrichment, which add essential metadata and external insights; and verification to ensure quality and compliance. Optional steps like signing and consolidation enhance authenticity, integrity, and scalability, and the transportation and analysis phases focus on secure distribution and leveraging SBOMs for compliance, security, and dependency analysis.

What constitutes a good SBOM generation process?

Format Agnostic

A robust SBOM generation process should support multiple SBOM formats, and the tiger team limited its scope to SPDX and CycloneDX. These widely recognized formats ensure compatibility with a variety of tools and stakeholders, facilitating broader adoption and usability. While the tiger team prioritized format flexibility within this scope, it treated JSON as the universal output format for all workflows, enabling consistent and standardized processing regardless of the target SBOM format.

Meets NTIA Minimum Elements

Adherence to NTIA's Minimum Elements for SBOM ensures that essential data points for the SBOM and each component are captured. These data points include:

- SBOM data points
    - Timestamp
    - Author Name and Contact Information
- Data points for each component
    - Supplier Name
    - Component Name
    - Version String
    - Unique Identifiers
    - Dependency Relationships

In addition, we extended the NTIA Minimum Elements to include License Information for all components in the SBOM (part of the 'Enrichment' process defined below). We note that these

[4] See SBOM Generation Reference Implementation Lifecycle Documentation

Minimum Elements may be further updated by the Cybersecurity and Infrastructure Security Agency (CISA).

Meets Framing Software Component Transparency (3rd edition)

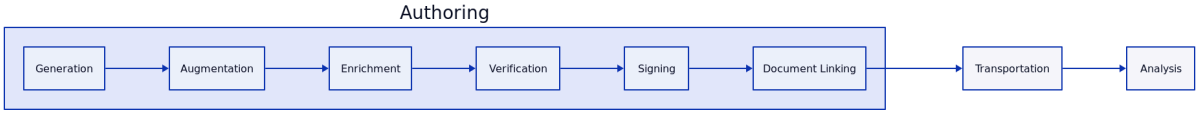
In addition to meeting the NTIA Minimum Elements, we also set out to meet the 3rd edition of the Framing Software Component Transparency (3rd edition)[5].

Automated Generation

A cornerstone of an effective SBOM generation process is automation. The tiger team strongly advocates integrating this process into the CI/CD pipeline, ensuring that SBOMs are automatically generated and delivered as part of the software development lifecycle.[6] This approach guarantees consistency, minimizes manual effort, and seamlessly incorporates SBOM generation into existing workflows.

SBOM Producer Lifecycle

The SBOM Producer Lifecycle is a framework for generating, augmenting, and sharing SBOMs effectively. This lifecycle is format-agnostic and complementary to guidance from NTIA and CISA. These steps may or may not be completed by separate tools. Current SBOM tools would benefit greatly from allowing NTIA-mandated metadata to be injected directly during the generation process. The SBOM Producer Lifecycle was developed while creating the SBOM Generation Reference Implementations, but it is not the first to capture similar phases. For example, OWASP documents a similar lifecycle in the Authoritative Guide to SBOM.



Authoring Stages

Generation

The generation phase produces the initial SBOM, capturing components, dependencies, and associated metadata. The completeness of this phase depends on factors such as the programming language, the package manager for said language, the capabilities of the tools used, and whether the SBOM is generated from source code prior to build time (Source SBOM)

[5] See Framing Software Component Transparency (3rd edition)
[6] This aligns with SBOM Sharing Primer

or during the build process (Build SBOM). Build SBOMs often produce more precise and comprehensive information.

Augmentation

This phase adds information that generally cannot be auto-generated, such as:

- Author details
- Supplier data
- Licensing information
- Copyright
- Additional identifiers including hashes
- Relationships

Augmentation helps ensure compliance with NTIA Minimum Elements.

Enrichment

Enrichment involves supplementing SBOMs with data from external sources, such as package repositories or open datasets. Examples of open datasets include ClearlyDefined and Ecosyste.ms. This step addresses missing details about components and licensing.

Verification

A minimum verification would be to ensure the resultant SBOM meets the format requirements for the standard. This can be done by running utilities or verifying against schemas provided by the format standard community. Additional verification can include semantic verification , verifying the SBOM meets the minimum requirements, and enhances SBOM quality.

Signing (Optional)

SBOM signing enables downstream consumers to verify authenticity and integrity, providing a way to verify that the SBOM has not been tampered with and originates from a trusted source. Ideally, this process is integrated into the CI/CD pipeline, utilizing cryptographic methods such as digital signatures or cryptographic checksums. Tools like Sigstore can be leveraged for automated signing workflows, while solutions such as in-toto can provide a complete attestation chain, ensuring traceability and security throughout the software supply chain.

Document Linking (Optional)

Even for a small SBOM implementation, you are likely to generate dozens of SBOMs across containers and various language stacks. We recommend structuring them hierarchically instead of merging them into a single consolidated SBOM. For example, microservices can be grouped into their own SBOMs, forming intermediate layers that are then tied together into a top-level SBOM. This hierarchical approach retains critical context about where components reside (e.g., which services or microservices they belong to) while providing a structured view of

dependencies. The `externalReferences` in CycloneDX, `external document reference` in SPDX version 2.X and `SpdxDocument import` in SPDX version 3.X can be used to link SBOMs. This method ensures clarity, scalability, and maintainability across complex software stacks.

Note that Consolidation could also be done in the Transportation stack.

Transportation

Transportation distributes SBOMs to stakeholders using secure methods[7]. Emerging standards, such as the proposed Ecma International TC54 Transparency Exchange API (TEA)[8], aim to streamline SBOM sharing.

Analysis

This stage leverages SBOMs for compliance and security purposes. Use cases include vulnerability scanning, license compliance, and dependency analysis.

Tool Selection Criteria

To support the implementation of our framework, we established clear selection criteria for evaluating tools used at each phase of the process. The tools were categorized and prioritized as follows:

1. **Preferred: Open-source tools actively maintained by a diverse group of contributors**
    - These tools are developed and supported by foundations or independent communities with multiple active contributors, rather than primarily driven by a single individual.
    - They must demonstrate consistent updates and long-term support, ensuring they are not one-off or abandoned projects.
2. **Second choice: Open-source tools backed by commercial vendors**
    - These tools are open-source but supported by commercial entities, providing additional resources and support while introducing some level of vendor dependence.
3. **Second choice: Open-source tools with a small number of highly active maintainers**
    - These tools are developed by a small team of maintainers who provide excellent support, but the projects' longevity is uncertain.
4. **Last resort: Proprietary tools**
    - These are commercial, closed-source tools that were considered only when options from the first two categories were insufficient.

[7] See SBOM Sharing Primer
[8] See Transparency Exchange API Specification

Our focus was primarily on tools from the first and second categories, as they best aligned with our principles of openness, transparency, and long-term sustainability. It is important to note that the authoring stages are tool-agnostic, allowing tools to be swapped out and replaced with alternatives without disrupting the overall process.

Tools Used in Our Example Workflow

Tool	License	Category	Link
Trivy	Apache 2.0	Generation	github.com/aquasecurity/trivy
sbomasm	Apache 2.0	Augmentation	github.com/interlynk-io/sbomasm
Parlay	Apache 2.0	Enrichment	github.com/snyk/parlay
sbomqs	Apache 2.0	Verification	github.com/interlynk-io/sbomqs

For additional tools, we recommend consulting one of the many SBOM tool databases, including:

- OpenSSF's SBOM Landscape
- CycloneDX's Tool Center
- SPDX's Tools

Recommendations

Tooling Improvements

Consolidation of Stages

Current SBOM tools would benefit greatly from allowing NTIA-mandated metadata to be injected directly during the generation process. Features such as command-line arguments or configuration-based options could enable users to provide this critical information upfront. This approach would effectively merge the augmentation phase into the generation phase, simplifying workflows and reducing the risk of errors associated with separate enrichment steps.

As tooling evolves, we anticipate that this capability will become a standard feature. Mature ecosystems and tools will streamline SBOM generation by automating the inclusion of essential metadata, making the process faster, more reliable, and fully compliant with NTIA requirements.

Open Source Supplier Identification

Supplier identification is a foundational element of an SBOM, yet many tools struggle to provide consistent and accurate data in this area. To improve supplier accuracy, SBOM tools must prioritize embedding NTIA-compliant metadata directly into their outputs. Enhancing open datasets like ecosyste.ms can play a crucial role by offering reliable mappings between components and their respective suppliers.

Collaboration with package ecosystems (e.g., npm, PyPI, Maven) is equally important to ensure NTIA-adherent metadata is included directly in package configurations. Embedding supplier details at the source reduces the need for post-processing and manual corrections in SBOM workflows. It should be noted, however, that many package ecosystems are actively undergoing improvements in this domain, with efforts to standardize and enrich metadata already underway. These developments, coupled with enhanced tools and workflows, will provide a stronger foundation for generating reliable, compliant SBOMs that meet modern supply chain security demands.

License Consistency

A common challenge in SBOM generation is the handling of license data. Most SBOM generation tools will indiscriminately include any license information they encounter, which often leads to inconsistencies and inaccuracies. For example, it is not uncommon to find duplicate licenses, such as 'Apache-2.0' and 'Apache 2.0', or even licenses that do not conform to SPDX license expressions. These discrepancies can create significant downstream issues for compliance and analysis.

To address this, it is crucial to implement robust verification and cleanup processes. Tools like Parlay can help ensure valid mappings between OSI licenses and SPDX identifiers, serving as a first line of defense against invalid or inconsistent license data. Additionally, cleanup scripts should be incorporated into the workflow to normalize license entries, remove duplicates, and flag non-compliant licenses for further review. By addressing these mismatches early in the process, organizations can maintain higher data quality in their SBOMs, ensuring accuracy and compliance while reducing manual intervention.

Benchmarking Completeness

Define clear metrics for SBOM completeness. While the NTIA Minimum Elements provide a baseline, tools such as the SPDX NTIA Checker can assist in evaluating adherence.

Ecosystem Challenges

- **Single Points of Failure:** Reliance on single providers, such as ecosyste.ms, introduces risks. Decentralizing metadata sources and verifying data integrity are critical.
- **Unknown Unknowns:** SBOMs often omit critical information. Tools should flag potential gaps and encourage manual review for completeness.
- **Inconsistent Data Mapping:** No authoritative mapping exists between fields in SPDX and CycloneDX. Establishing standardized mappings will improve interoperability.

- **Ambiguous Identifiers:** Multiple identifier standards exist without clear mapping between them and they may not uniquely identify a component[9]. For example, CPE may not be available or CPE documented may not match to all known CVEs for components, other unique identifiers may not have ways to be matched to vulnerabilities.
- **Self-Inclusion of Tools:** SBOM generation tools should include themselves in the resulting SBOM. Auditing existing tools for this capability can identify gaps.

Language-Specific Gaps

C/C++ SBOM Generation

SBOM tooling for C/C++ projects is scarce. These projects fall into two categories:

- **Single Image Builds** (e.g., ELF binaries): SBOMs should detail all linked libraries and dependencies.
- **Application-Level Builds**: Focus on capturing both build and runtime dependencies.

Non-Package Manager Dependencies

Many ecosystems lack package managers, complicating SBOM generation. Building metadata support at the distribution level (e.g., APT, YUM) can bridge this gap.

Collaboration Between Build and Distribution Systems

Build systems should generate metadata during the build process, while distribution systems maintain dynamic package information. Collaborative efforts can improve dependency tracking and SBOM accuracy.

Conclusion

SBOMs are pivotal for software transparency, security, and compliance. While progress has been made, significant challenges remain in achieving complete, accurate, and interoperable SBOMs. By addressing gaps in tooling, standardization, and ecosystem collaboration, the software industry can build a resilient and efficient SBOM generation process.

This work is not meant to be a continuous standing effort, but further attempts to expand and improve on this may make sense as technology and practices improve.

References

[9] See Software Identification Ecosystem Option Analysis

- Authoritative Guide to SBOM
- NTIA Minimum Elements
- CISA SBOM Sharing Lifecycle Report
- SPDX NTIA Checker
- CycloneDX Transparency Exchange API

Acknowledgments

The SBOM Generation Reference Implementations and this white paper were drafted and revised by a diverse set of experts from across the software ecosystem and are intended to capture a threshold of the current state of practice and expectations for software transparency in 2025.

The leaders of the SBOM Reference Implementation Tiger Team, Ian Dunbar-Hall (Lockheed Martin) and Viktor Petersson (sbomify), thank the members of the Tiger Team for their time, expertise, and dedication to the content presented in this document's third edition. We also thank our hosts from CISA, namely Allan Friedman, Victoria Ontiveros, and Jeremiah Stoddard, for their assistance in bringing this tiger team together and hosting us as we worked through and finalized the content.

Acknowledgments do not imply endorsement of this document and its content.

- Adolfo García Veytia
- Daniel Moch, Lockheed Martin
- Doug Dennis, PNNL
- Gary O'Neall, Source Auditor and SPDX
- Ian Dunbar-Hall, Lockheed Martin
- Manoj Prasad, Faculty, Computer Science, Western Washington University
- Ricardo A Reyes, Chainguard
- Viktor Petersson, sbomify
- Tieg Zaharia, Tidelift
- Duncan Sparrell, sFractal
- Przemysław Roguski, Red Hat
- Melissa Rhodes, Medtronic
- John Cavanaugh, Internet Infrastructure Services Corp.Scott Van Eps, Danaher