

TLP - CLEAR



EUROPEAN UNION AGENCY
FOR CYBERSECURITY

ENISA Technical Advisory for Software Bill of Materials

OCTOBER 2026

About ENISA

The European Union Agency for Cybersecurity, ENISA, is the Union's agency dedicated to achieving a high common level of cybersecurity across Europe. Established in 2004 and strengthened by the EU Cybersecurity Act, the European Union Agency for Cybersecurity contributes to EU cyber policy, enhances the trustworthiness of ICT products, services and processes with cybersecurity certification schemes, cooperates with Member States and EU bodies, and helps Europe prepare for the cyber challenges of tomorrow. Through knowledge sharing, capacity building and awareness raising, the Agency works together with its key stakeholders to strengthen trust in the connected economy, to boost resilience of the Union's infrastructure, and, ultimately, to keep Europe's society and citizens digitally secure. More information about ENISA and its work can be found here: www.enisa.europa.eu.

CONTACT

To contact the authors, use product_security@enisa.europa.eu.

For media enquiries about this paper, use press@enisa.europa.eu.

AUTHOR

ENISA

LEGAL NOTICE

This publication represents the views and interpretations of ENISA, unless stated otherwise. It does not endorse a regulatory obligation of ENISA or of ENISA bodies pursuant to Regulation (EU) 2019/881.

ENISA has the right to alter, update or remove the publication or any of its contents. It is intended for information purposes only and must be accessible free of charge. All references to it or its use as a whole or in part must contain ENISA as its source.

Third-party sources are quoted as appropriate. ENISA is not responsible or liable for the content of the external sources, including external websites, referenced in this publication.

Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

ENISA maintains its intellectual property rights in relation to this publication.

COPYRIGHT NOTICE

© European Union Agency for Cybersecurity (ENISA), 2026

Unless otherwise noted, the reuse of this document is authorised under the Creative Commons Attribution 4.0 International (CC BY 4.0) licence (<https://creativecommons.org/licenses/by/4.0/>). This means that reuse is allowed, provided appropriate credit is given and any changes are indicated.

Copyright for the image on the cover: © Adobe Stock

For any use or reproduction of photos or other material that is not under the ENISA copyright, permission must be sought directly from the copyright holders.

ISBN 978-92-9204-813-6, ISSN 3094-600X doi: 10.2824/2077129, Catalogue number TP-01-26-022-EN-N

Disclaimer: Generative AI was used in a limited capacity to support language refinement and preliminary document screening. All outputs were reviewed and validated by subject-matter experts. No AI generated content was used without substantive human oversight.

Table of contents

About ENISA	1
1. Introduction	3
1.1 Introduction	3
1.2 Scope and Context	4
1.3 Note on the current version of the document	4
2. SBOM Architecture and Data Flows	6
2.1 How Software Bill of Materials works	6
2.1.1 Data Model	7
2.1.2 SBOM Formats	10
2.1.3 SBOM Types	11
2.2 SBOM Lifecycle	13
2.2.1 SBOM Lifecycle Phases	14
2.2.2 SBOM as part of a CI/CD Pipeline	15
2.2.3 SBOM Open-source Tooling Landscape	17
3. SBOM Threats and Risks	19
3.1 SBOM Threats	19
3.2 SBOM General Risks	21
4. SBOM Best Practices	24
4.1 SBOM Integration and Automation	24
4.1.1 Integration Patterns	24
4.1.2 Common Configuration Requirements	25
4.2 General Security Recommendations	26
5. Bibliography	28

1. Introduction

1.1 Introduction

Modern software products contain hundreds or thousands of components, each potentially harbouring vulnerabilities that threaten application and operating system security. Without component visibility, organisations operate blindly, discovering exposures only after exploitation occurs.

The **Software Bill of Materials (SBOM)** supports proactive vulnerability tracking by identifying components with potential vulnerabilities, allowing security teams to correlate component inventories against publicly disclosed security vulnerabilities and respond to emerging risks. Beyond immediate vulnerabilities, SBOM data supports the assessment of component viability and supply chain sustainability risks.

An **SBOM** is a formal record containing details and supply chain relationships of components included in the software elements of a product with digital elements. The SBOM should be drawn in a commonly used and machine-readable format identifying and documenting components contained in the products with digital elements. It can provide those who manufacture, purchase, and operate software with information that enhances their understanding of the supply chain, which has multiple benefits, in particular it helps manufacturers and users to track known newly emerged vulnerabilities and cybersecurity risks ⁽¹⁾.

Core Capabilities

- **Transparency.** Reveals third-party and open-source components comprising modern software.
- **Traceability.** Links deployed software to upstream suppliers and downstream consumers.
- **Compliance.** Provides auditable records for regulatory and contractual requirements.
- **Security.** Supports vulnerability detection and software provenance verification.

SBOMs form a foundational data layer rather than a complete security solution. Their value emerges through integration with security processes such as timely patching, mitigation activities, and documented vulnerability handling decisions, with continuous maintenance, and systematic deployment across the software development lifecycle (SDLC).

This document provides practical guidance on the adoption of SBOM, and it is intended for organizations especially micro-, small and medium enterprises (SMEs) and stewards seeking to set up the initial capabilities to enhance the software supply chain transparency. It aligns with the security principles from the ENISA Secure by Design and Default Playbook ⁽²⁾, particularly supply chain controls, lifecycle management, and secure coding and verification practices. It introduces key concepts, industry standards, use cases and implementation practices to support organisations initiating their SBOM journey. In particular, this document:

¹ Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act) (OJ L, 2024/2847, 20.11.2024, ELI: <http://data.europa.eu/eli/reg/2024/2847/oj>).

² ENISA, Secure by Design and Default Playbook, available at <https://www.enisa.europa.eu/publications/enisa-secure-by-design-and-default-playbook>

- outlines the SBOM implementation architecture and data flows based on a defined lifecycle;
- presents an indicative SBOM threat model along with targeted risks;
- lists measures to address the security issues across the SBOM ecosystem.

This technical advisory considers regulatory requirements such as those set out in the Cyber Resilience Act (CRA) ⁽³⁾, in particular essential requirements related to SBOM obligations for manufacturers but essentially describes recommended best practices that may go beyond those requirements.

It aims to provide technical context to such requirements by describing common threats across the SBOM lifecycle along with practical recommendations to mitigate them. However, this document should not be considered as a legal interpretation of the CRA.

1.2 Scope and Context

This technical advisory is intended for guidance purposes only. The recommendations and best practices described herein are non-mandatory and do not create or modify any legal, regulatory, or compliance obligations.

The scope of this technical advisory is limited to Software Bill of Materials and does not cover other emerging Bill of Materials (BOM), such as Software Bill of Materials for AI, Hardware Bill of Materials (HBOM), Cryptographic Bill of Materials (CBOM) or other variants. While all artifacts may share some common principles, they also have differentiators and therefore are considered out of scope for this advisory.

This technical advisory remains sector-agnostic and applicable across all types of organisations regardless of their operational environment or industry. It is primarily focused on organisations that are at the early stage of SBOM adoption, and it does not provide a framework for assessing and improving organisational maturity. Future editions may cover the topic more extensively.

1.3 Note on the current version of the document

The current version of the document is released as a result of the ENISA call for feedback ⁽⁴⁾ which took place between the 17th of December 2025 and the 23rd of January 2026 and it is heavily influenced by the survey results presented at ENISA's SBOM Adoption State of Play 2026 published on the 9th of June 2026 ⁽⁵⁾. Via the call for feedback ENISA received 67 contributions from public and private stakeholders, cybersecurity experts, and the open-source community.

Based on this feedback and the survey results, ENISA recognised the need to transform the “**SBOM Landscape Analysis: Towards an implementation guide**” into a “**Technical Advisory for Software Bills of Materials**” to better respond to the needs and feedback of the ecosystem.

³ Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act) (OJ L, 2024/2847, 20.11.2024, ELI: <http://data.europa.eu/eli/reg/2024/2847/oj>).

⁴ ENISA, 'Call for feedback: Advancing software supply chain security together!', press release, 17 December 2025.

⁵ ENISA, SBOM Adoption State of Play – 2026, available at <https://www.enisa.europa.eu/publications/sbom-adoption-state-of-play-2026>

Finally, this version also reflects the feedback provided by **experts participating in the SBOM Workstrand under the CRA Expert Group** as part of the review cycle conducted between 31 July and 11 September 2026.

Acknowledgements

We would like to publicly thank all stakeholders that provided their views and feedback! All contributions have been reviewed and, to the maximum extent possible, the recommendations and observations have been addressed in this final version. This document may go through regular revisions.

2. SBOM Architecture and Data Flows

2.1 How Software Bill of Materials works

An SBOM is a structured inventory that captures software components and their dependencies for a given software product. Creating this "list of ingredients" gives manufacturers and users a better understanding and greater visibility into the "ingredients" of their software product, who produced them, where they came from and whether those "ingredients" may pose any risks.

The **SBOM generators gather information** from different sources in order to effectively identify first-party components, open-source libraries, third-party commercial libraries, operating system packages, container images etc. Indicative **sources of information** are:

- source code repositories,
- build artifacts,
- package manager manifests,
- lock files,
- container images,
- binary analysis.

For each of the identified software components, the SBOM generator then collects and correlates **metadata** such as component name, component version, component creator, unique identifiers, license information, external references and dependencies. The scope and completeness of the collected metadata depend on its availability, the SBOM generator's analysis methods, and the selected SBOM data model (e.g. CycloneDX or Software Package Data Exchange (SPDX)). The result is not merely a list of components but also a representation of their dependency relationships.

Upon the completion of the software composition analysis, the metadata collection and relationship discovery, an SBOM is generated in a **standard machine-readable format**, most commonly CycloneDX or SPDX. The SBOM can be generated and stored as JSON, XML, YAML or Tag-Value. It is important to highlight that SBOMs can be generated in different phases of the Software Development Lifecycle (SDLC). SBOMs generated in different SDLC phases serve different needs and contain different information.

The generation of an SBOM provides limited value unless it is **actively consumed** to support operational and security processes. The benefit of an SBOM lies in its **integration** into use cases such as vulnerability management, software asset inventory, license compliance, procurement, and regulatory compliance.

The need to treat an SBOM as more than a compliance artifact, is also reflected in the findings of the **ENISA SBOM Adoption State of Play 2026**, as respondents identified **industry consensus on best practices** for integrating SBOM production and consumption into software development practices as the highest priority (23%), followed by best practices for integrating SBOMs into risk and compliance processes (19%) and guidance on how SBOM production and consumption methods should evolve over time (19%). These findings highlight that organisations consider the **operational integration of SBOMs** into existing development, security, and governance workflows as a crucial step for a mature SBOM adoption.

2.1.1 Data Model

With the aim to enable interoperability and consistency throughout the SBOM generation and consumption, authoritative bodies have published guidance on the data fields for SBOMs. The **German Federal Office for Information Security (BSI)** has published a technical guideline defining the required data fields, while the **U.S. Cybersecurity and Infrastructure Security Agency (CISA)**, building upon the existing work previously published by the National Telecommunications and Information Administration (NTIA), defines the “Minimum Elements for a Software Bill of Materials”. Both publications provide a baseline for generating SBOMs.

The table below attempts to map the BSI required data fields with the U.S. CISA defined minimum elements in an effort to support organisations in producing SBOMs that satisfy international interoperability and regulatory requirements.

Data Fields	Technical Guideline BSI TR-03183: Part 2 Software Bill of Materials ⁽⁶⁾	Data Fields	U.S. CISA “Minimum Elements for a Software Bill of Materials (SBOM)” ⁽⁷⁾
SBOM Metadata			
Creator of the SBOM	Email address of the entity that created the SBOM. If no email address is available this MUST be a “Uniform Resource Locator (URL)”, e.g. the creator’s home page or the project’s web page.	SBOM Author	The name of the entity that creates the SBOM data for the primary component.
-	-	SBOM Version	Identifier designated by the <i>SBOM Author</i> to specify a change in the SBOM document from a previously identified version or to indicate that it is the first version.
SBOM-URI	“Uniform Resource Identifier (URI)” of this SBOM.	-	-
-	-	SBOM Data Format Name	The name of the data format used to represent the SBOM data.
-	-	SBOM Data Format Version	Identifier designated by the SBOM data format to specify the version of the data format.
-	-	SBOM Author Signature	A digital signature attributable to the <i>SBOM Author</i> .
-	-	SBOM Tool Name	The name of the tool used by the <i>SBOM Author</i> to generate or amend the SBOM.

⁶ BSI, Technical Guideline BSI TR-03183: Cyber Resilience Requirements for Manufacturers and Products Part 2: Software Bill of Materials, available at https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03183/BSI-TR-03183-2_v2_1_0.html

⁷ CISA, 2026 Minimum Elements for a Software Bill of Materials (SBOM), available at <https://www.cisa.gov/resources-tools/resources/2026-minimum-elements-software-bill-materials-sbom>

-	-	SBOM Tool Version	Identifier for the version of the tool identified in the <i>SBOM Tool Name</i> element.
Timestamp	Date and time of the SBOM data compilation according to the specification of the formats. Note: It is recommended to only use timestamps in UTC.	SBOM Timestamp	Record of the date and time of the most recent update to the SBOM data.
-	-	SBOM Generation Context	The relative software lifecycle phase and data available at the time the <i>SBOM Author</i> generated the SBOM (before build, during build, after build).
Component Data			
Component creator	Email address of the entity that created and, if applicable, maintains the respective component. If no email address is available this MUST be a "Uniform Resource Locator (URL)".	Component Producer	The name of an entity that creates, defines, and identifies components.
Component name	Name assigned to the component by the component creator. If no name is assigned this MUST be the actual filename.	Component Name	The name assigned by the <i>Component Producer</i> to a software component.
Filename of the component	The actual filename of the component.		
Component version	Identifier used by the creator to specify changes in the component to a previously created version.	Component Version	Identifier used by the <i>Component Creator</i> to specify a change in a software component from a previously identified version or to indicate that it is the first version.
Source code URI	"Uniform Resource Identifier (URI)" of the source code of the component, e.g. the URL of the utilised source code version in its repository, or if a version cannot be specified the utilised source code repository itself.	Component Identifiers	Identifier(s) used to identify a component or serve as a look-up key for relevant databases.
URI of the deployable form of the component	"Uniform Resource Identifier (URI)", which points directly to the deployable form of the component.		
Other unique identifiers	Other identifiers that can be used to identify the component or to look it up in relevant databases, such as Common Platform Enumeration (CPE) or Package URL (purl).		
URL of the security.txt (optional)	Contains the "Uniform Resource Locator (URL)" of the component creator's security.txt.		

Hash value of the deployable component	Cryptographically secure checksum (hash value) of the deployed/ deployable component.	Component Hash Value	The output generated from applying a cryptographic hash algorithm to an executable component artifact.
Hash value of the source code of the component (optional)	Cryptographically secure checksum (hash value) of the component source code. A specific algorithm how to create a hash value of multiple source files or a source code tree, and which hash algorithm is utilised for that has not yet been determined.		
-	-	Component Hash Algorithm	The cryptographic algorithm used to compute the <i>Component Hash Value</i> of the software component.
Distribution licences	Distribution licence(s) of the component under which it can be used by a licensee.	Component License	The identifier(s) for the license(s) under which the software component is available.
Original licences	The licence(s) that have been assigned by the creator of the component.		
Effective license (optional)	The licence under which the component is used by the licensee that is the creator of the current SBOM.		
Dependencies on other components	Enumeration of all components on which this component is directly dependent, or which this component contains.	Component Dependency Relationship	The relationship between two components, given that one component is necessary for the operation of the other.
Executable property	Describes whether the component is executable; possible values are "executable" and "non-executable".	-	-
Archive property	Describes whether the component is an archive; possible values are "archive" and "no archive".	-	-
Structured property	Describes whether the component is a structured file, i.e. metadata of the contents is still present; possible values are "structured" and "unstructured".	-	-

Table 1 - Data Models

2.1.2 SBOM Formats

The SBOM format is also a determining factor for interoperability across the software supply chain. Standardized formats provide a common data model that enables software producers, consumers, security tools, vulnerability management platforms, and compliance solutions to process and exchange software composition analysis information in a consistent manner.

ENISA's SBOM Adoption State of Play 2026 mentions that almost **73%** of the organisations use CycloneDX or SPDX. Table 2 presents the two predominant formats based on their current version.

	SPDX (System Package Data Exchange) V 3.0.1 ⁽⁸⁾	CycloneDX V 1.7 ⁽⁹⁾
Description	Comprehensive framework for documenting system component information, with a historical focus on supporting detailed software metadata to enable tracking of vulnerabilities, licensing, and dependencies for product lines.	Security-focused approach to software bill of materials, emphasising vulnerability tracking, cryptographic integrity, and comprehensive dependency analysis.
Standard/Backing	SPDX® Specification v2.2.1 is an ISO-standardised format (ISO/IEC 5962:2021). SPDX 3. in process at ISO to be listed as an update to ISO/IEC 5962:2021. SPDX 3.0 has been ratified as an OMG standard ⁽¹⁰⁾ .	Ratified as ECMA-424 by ECMA International ⁽¹¹⁾ . Maintains a formal standardisation process and governance model through ECMA Technical Committee 54 (TC54). Currently pursuing ISO JTC1 FastTrack.
Key Requirement	Risk Analysis: Providing clear, machine-readable metadata to support product line risk analysis, including vulnerability management (native CVE & VEX metadata, CSAF and other formats, precise license compliance metadata, and dependency tracking), precise license compliance metadata, and dependency tracking.	Vulnerability Management: Enabling precise correlation between components and security vulnerabilities with native VEX (Vulnerability Exploitability eXchange) support.
	Supply Chain Transparency: Enabling comprehensive component provenance and cryptographic integrity, together with rich relationship options for creation of knowledge graph.	Security Automation: Facilitating seamless integration with security tools and CI/CD pipelines through lightweight, efficient formats.
	Regulatory Alignment: Meeting increasing demands for software transparency in regulated industries.	Operational Visibility: Documenting not just static components but runtime services, APIs, and operational dependencies.
	Interoperability: Establishing a common language for software	Cryptographic Integrity: Providing robust verification mechanisms for supply chain attestation and component authentication.

⁸ The Linux Foundation, The System Package Data Exchange® (SPDX®) Specification Version 3.0.1, available at <https://spdx.github.io/spdx-spec/v3.0.1/>

⁹ OWASP, Authoritative Guide to SBOM – Implement and optimize use of Software Bill of Materials, available at https://cyclonedx.org/guides/OWASP_CycloneDX-Authoritative-Guide-to-SBOM-en.pdf

¹⁰ OMG Standard, available at <https://www.omg.org/spec/SPDX/3.0/About-SPDX>

¹¹ EXCMA-424, available at <https://ecma-international.org/publications-and-standards/standards/ecma-424>

	composition across diverse toolchains and ecosystems.	
--	---	--

Table 2 - SBOM Formats

2.1.3 SBOM Types

It is important to understand that the selection of the SBOM format is not the only decisive factor influencing the information that the SBOM is going to contain. Based on “**when**” the SBOM is generated, it may contain different forms of information and serve different purposes. The choice of “when” to generate SBOMs should serve the operational needs of the Organization but also align with the applicable regulatory requirements.

In this case, the categorisation of SBOMs derives from the **phase of the SDLC** under which they are generated. The most recognised SBOM types based on the U.S. CISA types of SBOM ⁽¹²⁾ are:

SBOM Type	Definition	Information Potentially Captured	Indicative Use
Design	SBOM of intended design of included components (some of which may not exist) for a new software artifact.	- Planned third-party libraries - Open-source packages - Commercial components - External services and APIs - Expected versions - Intended licensing model	- Architectural decisions - Licencing purchase - Threat Modeling - Regulatory requirements management
Source	SBOM created directly from the development environment, source files, and included dependencies used to build a new product artifact.	- Existing components from the source code - Direct dependencies from third-party and open-source components - Exact versions - Licenses	- Dependency visibility - Early vulnerability detection
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.	- Direct dependencies - Transitive dependencies - Linked libraries - Generated artifacts - Build-time packages - Operating system components - Exact versions - Cryptographic hashes	- Regulatory compliance - Vulnerability Assessment - Software transparency

¹² U.S. CISA, Types of Software Bill of Materials (SBOM), available at <https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom>

		• Supplier Information • Dependency graph • Package URLs • CPE identifiers • Licenses	
Analysed	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.	• Direct Dependencies • Transitive dependencies • Linked Libraries • Generated artifacts • Versioning • Cryptographic information • Licenses • Integrity validation • Software origin • Supplier trust information	• License compliance • Provenance verification • Supply chain risk assessment • Security certification / verification
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.	• Installed packages • Container images • Environment-specific dependencies • Platform-specific libraries	• Enterprise-level asset inventory • Patch management • Environment-specific risk assessment • Configuration management
Runtime	SBOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an “Instrumented” or “Dynamic” SBOM.	• Dynamically loaded libraries • Plug-ins • Runtime modules • Container sidecars • Injected agents • Shared services	• Threat detection • Attack surface risk assessment • Runtime monitoring

Table 3 - SBOM Types

2.2 SBOM Lifecycle

Having established the key concepts of the SBOM data models, formats and types, this section presents the **SBOM lifecycle and architecture**. The SBOM provides added value when it is continuously managed throughout the SDLC and follows the software's evolution while components are added, updated, altered or deleted through new software releases.

A structured and automated SBOM lifecycle aims at ensuring that SBOMs are properly produced and consumed internally and externally. To highlight the need for this notion, a big gap was identified in the ENISA SBOM Adoption State of Play 2026, between what organisations **need from SBOMs** and what they **actually receive**. While 36 % of respondents stated that their organisations need all primary components and direct dependencies with declared known unknowns, only 29 % actually receive them. Furthermore, 24 % need SBOMs with full depth analysis, but only 14 % receive them. Finally, 27 % require SBOMs with depth analysis including declared known unknowns, yet only 12 % receive them.

This section presents the **SBOM lifecycle from both a conceptual and a technical perspective**. . It starts by illustrating the SBOM life-cycle phases from generation to consumption and continuous maintenance, then presents how these phases can be integrated and automated into a CI/CD pipeline and finally lists the available open-source tool stack to support all defined phases.

2.2.1 SBOM Lifecycle Phases

Figure 1 depicts **the conceptual perspective of the SBOM lifecycle** aligned with the key principles of SBOM. The lifecycle begins with the software build and SBOM generation based on the necessary sources of information, progresses to the SBOM validation to assess the quality of the generated SBOM, followed by cryptographic signing to protect its integrity and authenticity, secure storage, operational consumption and continuous maintenance and updates.

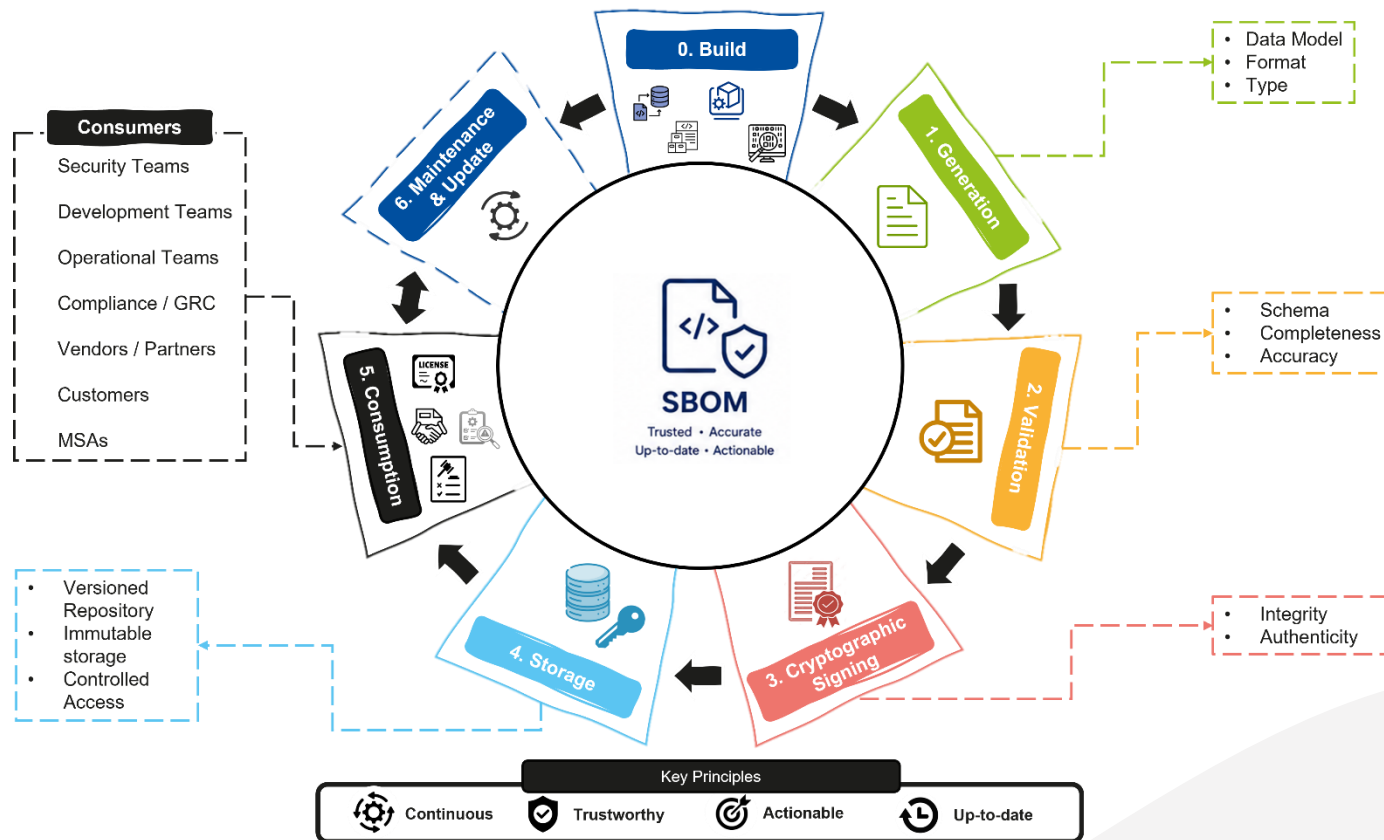


Figure 1 - SBOM Lifecycle

2.2.2 SBOM as part of a CI/CD Pipeline

The integration of SBOM generation and consumption into CI/CD pipelines is increasingly adopted and recognised as one of the best practises to achieve completeness, accuracy and trustworthiness. Automation enables organisations to generate, validate, update and consume up-to-date SBOMs. The findings of the SBOM Adoption State of Play 2026 reflect this, as 41 % of the organisations reported that they aim to **fully automate SBOM generation for each release/build**. However, automation is still a work in progress after the built stage as only 26 % of respondents believe that their organisations will fully automate SBOM updates across the product support period and almost 23 % believe that their organisations will partially automate this process.

While the previous section presented the conceptual perspective of the SBOM lifecycle, this section presents the **technical implementation of SBOM as part of a CI/CD pipeline**. The architectural diagram in Figure 2 aligned with the defined SBOM lifecycle phases depicts the integration of SBOM into the SDLC and the automation of each phase.

Each phase includes the key elements and processes to be considered successful along with basic failure points. This should not be interpreted as an exhaustive list of steps and aspects to consider when generating and consuming SBOMs but rather as an example of how SBOM integration and automation could be implemented.

The process begins with the source code build which produces the sources of information and then triggers the generation of SBOM. The SBOM generators attempt to identify the software components, extract relevant metadata, analyse dependencies, and produce machine-readable SBOMs in a standardized format (e.g., SPDX or CycloneDX). The generator collects, correlates, and structures metadata from the software build process and associated repositories.

Upon the successful generation of the SBOM, the validation process is triggered to assess if the SBOM is syntactically correct, conforms to the selected SBOM specification and contains the required metadata before triggering the next process. It is essential to also validate the component identifiers during this process.

Once validated, the SBOM is cryptographically signed to ensure authenticity and integrity before being stored in a repository or registry, to be consumed by downstream security tools.

The SBOM maintenance and update phase depicts the continuous process of keeping the SBOM up-to-date and synchronised with the current version of the software. Indicative automatic triggers are included to reflect the need to re-generate SBOMs and follow the lifecycle phases.

Finally, indicative security controls and trust mechanisms are depicted to protect the integrity, authenticity, and availability of SBOMs throughout the pipeline, enabling organisations to produce reliable SBOMs that can be consumed across development, security, operations, procurement and compliance teams.

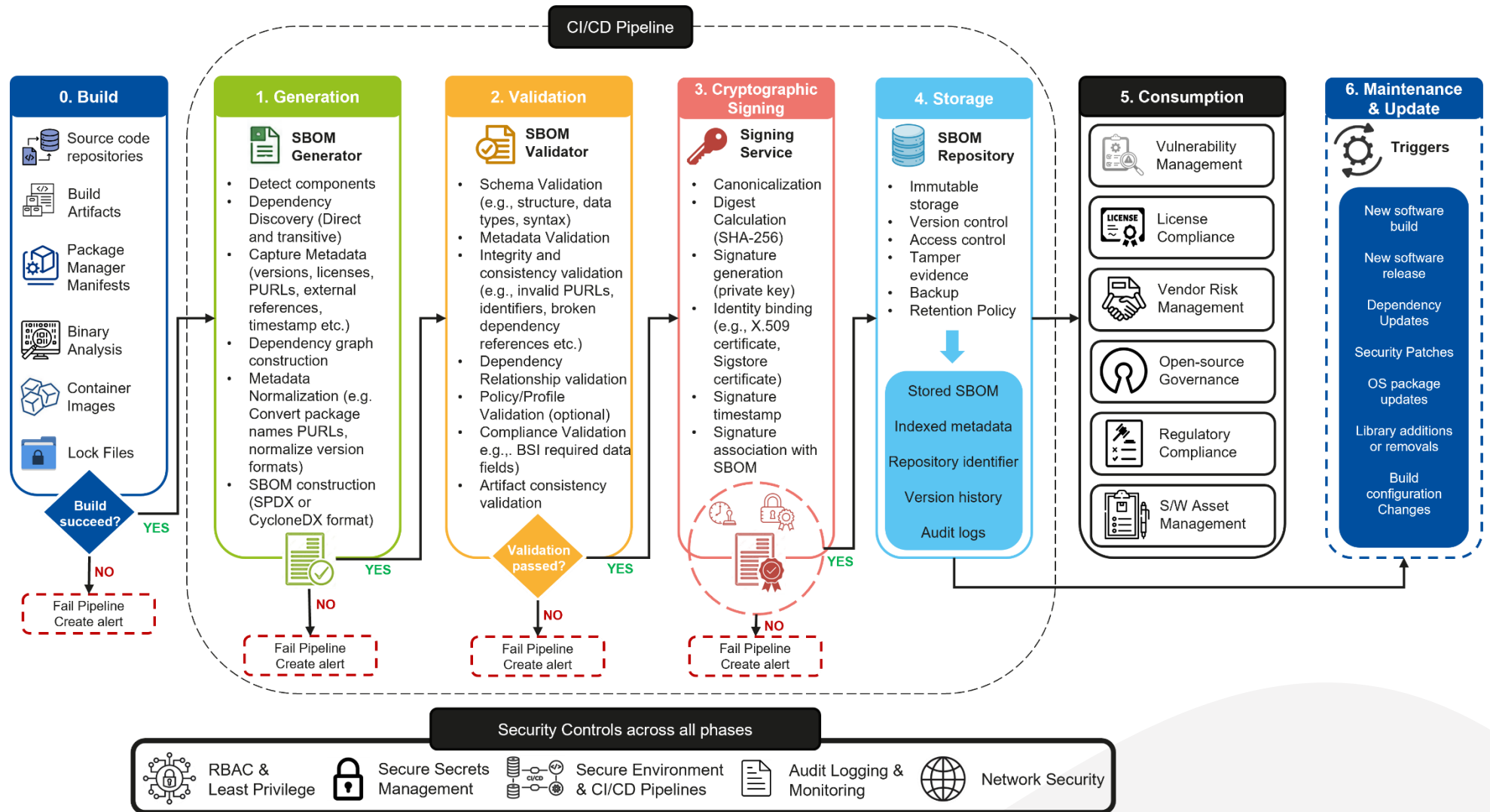


Figure 2 - SBOM Architectural Diagram

2.2.3 SBOM Open-source Tooling Landscape

Integration and automation are only possible when supported by the necessary tool stack. Based on the SBOM Adoption State of Play 2026, 33% of the organisations rely on open-source tools, thus Table 4 identifies open-source tools that could be integrated into a CI/CD pipeline to support the end-to-end SBOM lifecycle for both CycloneDX and SPDX formats. The tools are categorised based on the SBOM lifecycle phases and should not be considered as an exhaustive, but rather than an indicative list to support the purpose of this document.

The following table demonstrates that while some tools are format-agnostic and can support both SBOM formats, certain stages like SBOM generation, schema validation, and/or consumption require format-specific tooling.

SBOM Lifecycle Phase / Format	Generation	Validation		Cryptographic Signing		Store	Consumption
	Automation	Schema	Quality Score	Attestation	Keyless	Workflow artifact	Vulnerability Management
CycloneDX ⁽¹³⁾	cyclonedx-maven-plugin (Java/Maven), @cyclonedx/ cyclonedx-npm (npm), Cyclonedx-gomod (Go), Cyclonedx dotnet tool (.NET), cdxgen, Syft, SBOMit	CycloneDX CLI	sbomqs, sbom-scorecard (eBAY)	GitHub CLI (gh attestations generate)	Cosign, Sigstore	actions/upload-artifact@v4	Grype, Trivy, OWASP Dependency Check, OWASP Dependency-Track, GitHub Dependency Review, OWASP dep-scan, OSV-Scanner (Go), CVE Binary Tool,
SPDX ⁽¹⁴⁾	spdx-gradle-plugin, spdx-maven-plugin, spdx/tools-python, spdx/tools-java, spdx/tools-golang, yocto (enabled by default in poky), linux (make spdx), zephyr (west spdx).	tools.spdx.org/app/validate/ https://tools.spdx.org/app/ntia_checker/	sbomqs, sbom-scorecard (eBAY)	GitHub CLI (gh attestations generate)	Cosign, Sigstore	actions/upload-artifact@v4	Fossology, sbomify, sbom-cve-check, tools.spdx.org/app/spdx3_viz, GitHub Dependency Review

¹³ <https://cyclonedx.org/tool-center/>

¹⁴ <https://spdx.dev/use/spdx-tools/>

kubernetes-sigs/bom, GitHub SBOM, Microsoft sbom-tool (.NET), Syft, Trivy, ORT, SBOMit							
---	--	--	--	--	--	--	--

Table 4 - Open-source SBOM Tool Stack

The fragmentation of SBOM formats poses a significant challenge to the implementation and tooling landscape. Tooling exists for seamless conversion between the formats, such as OpenSSF's ⁽¹⁵⁾ Protobom.

¹⁵ <https://sbom-catalog.openssf.org/catalog/#/Circle/Normalize/HowTo>

3. SBOM Threats and Risks

3.1 SBOM Threats

This section presents an **SBOM focused STRIDE threat model** to support organisations in the identification and mitigation of threats across the SBOM lifecycle. The STRIDE threat model, developed by Microsoft, provides a structured framework for the continuous assessment of threats. It is divided in six categories: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privilege.

The identified threats not only focus on the SBOM artifact but rather on the entire SBOM ecosystem (e.g., SBOM generation tools, validation and signing service, repositories, consumption pipelines) (Table 5).

STRIDE	Threat	Threat Scenario	Potential Impact	Indicative Mitigation Activities
Spoofing	Spoofed Identity of SBOM Author or Software Producer	Impersonation of a legitimate SBOM Author or Software producer to distribute compromised SBOM.	Trust and use malicious software or compromised components.	- Digital signatures - Supplier identity verification
	Spoofed Consumer Identity	Impersonation of a legitimate consumer to gain unauthorised access to restricted SBOM.	Gain access to restricted information that assist attacker reconnaissance and targeted attacks.	- Consumer identity verification - Controlled SBOM sharing
	Spoofed CI/CD pipelines	SBOM appears to be generated from a trusted CI/CD pipeline.	Trust in malicious software provenance.	- Build attestations - Build Attestation - Authenticated CI/CD pipelines
	Spoofed SBOM generation tool	A malicious tool generates SBOM that appear to be valid.	Trust and use malicious or compromised SBOM.	- Trusted tool use - Code signing
	Spoofed certificates	Stolen certificates are used to sign malicious SBOM.	Trust and use malicious SBOM appearing to be legitimate from a trusted source.	- Secure key Management - Certificate revocation
	Spoofed SBOM repository mirror	Compromised repository mirror distributes malicious SBOM appearing to be legitimate.	Trust and use malicious SBOM appearing to be legitimate from a trusted source.	- SBOM Signature verification - Compare cryptographic hashes against trusted metadata - Monitor repository certificates

Tampering	Tampered SBOM data	Components, versions or dependency relationships are tampered (added, deleted, or altered) after generation.	Incorrect dependency analysis and vulnerability identification.	- Integrity verification - Signature verification
	Tampered metadata	Component versions, hashes, licenses or suppliers are tampered before SBOM generation.	Inaccurate software inventories and compliance breach.	- Cryptographic integrity check
	Tampered hashes	Cryptographic hashes are altered.	Lack of proper artifact verification.	- Document Signature
	Tampered provenance	Build provenance is altered.	Trust and use malicious SBOM appearing to be from a trusted source.	- Signed attestations
	Tampered external references	PURLs or CPEs are altered.	Use and act upon compromised resources appearing as known and trusted to conduct vulnerability matching and assessment.	- Digital signatures - External references verification
Repudiation	Software Producer or SBOM Author repudiation	Software Producer or SBOM Author denies the generation or publication of SBOM.	Lack of accountability.	- Digital signatures - Timestamps - Audit logs
	Distribution repudiation	SBOM Author or Software producer denies the distribution of an SBOM.	Reduced supply chain transparency.	- Repository logging - Signed releases
Information Disclosure	Technology stack disclosure	Lack of controlled SBOM sharing reveals software frameworks and versions.	Assist attacker to conduct reconnaissance and increase attack surface.	- Controlled SBOM sharing - Access control
	Component disclosure	Proprietary software components become visible.	Intellectual property rights breach.	- Controlled SBOM sharing - Access control
	Supplier relationship disclosure	Supplier relationships become publicly visible.	Increased attack surface and more targeted attacks.	- Controlled SBOM sharing - Access control
	Dependency graph disclosure	Software architecture and dependency relationships become publicly visible.	Increased attack surface and more targeted attacks.	- Controlled SBOM sharing - Access control
	Version disclosure	Exact software versions enable vulnerability mapping.	Faster exploitation using known CVEs.	- Controlled SBOM sharing - Access control

	Cryptographic material / metadata disclosure	Embedded certificates or key identifiers leak trust relationships. Hashes or identifiers reveal internal release processes.	Assist attacker to conduct reconnaissance and increase attack surface.	- Key management - Access control - Audit Logging
Denial of Service	SBOM repository DoS	SBOM repositories are not available.	SBOM unavailable during incident response or procurement.	- Redundant infrastructure - DDoS protection
	Parser DoS	Extremely large or deeply nested SBOM cause excessive computation and exhaust parser resources.	Resource exhaustion and parser failures.	- SBOM size limitation - Parser hardening
	Storage DoS	Large number of SBOM exhaust repository capacity.	Resource exhaustion and potential disruption of the CI/CD pipelines if SBOM is a mandatory step.	- SBOM size and number limitation. - SBOM Retention Policy
Elevation of Privilege	CI/CD pipeline privilege escalation	Elevated privileges in the build pipeline generating SBOMs.	Corrupted SBOM generation that appears to be legitimate.	- Access control - CI/CD pipeline hardening
	SBOM repository privilege escalation	Unauthorised access to SBOM repository.	Use and distribution of compromised SBOM.	- Access control
	Signing Infrastructure privilege escalation	Compromised infrastructure creates trusted SBOM.	Use and trust of compromised SBOM.	- Key management - Access control

Table 5 - SBOM Threat Model

3.2 SBOM General Risks

Complementary to the identification of threats, this section presents some general risks that jeopardise the SBOM accuracy, integrity and operationalisation. Table 6 lists indicative risk scenarios, their potential impact and suggested mitigation activities.

Risk	Risk Scenario	Potential impact	Indicative Mitigation Activities
Non-standardised SBOM Format	SBOMs generated using a proprietary, custom, unstructured, or otherwise non-interoperable format.	Reduced interoperability between producers and consumers.	- Generate SBOMs using established standards, such as SPDX or CycloneDX
SBOM in a Non-Machine-Readable Format	SBOMs provided only in a format intended primarily for human consumption.	Reduced automation and scalability of SBOM consumption and inability to automatically correlate components	- SBOMs in a machine-readable format suitable for automated processing

		with vulnerability databases.	
Incomplete SBOM	Components or dependencies are not fully covered by the SBOM.	Vulnerabilities remain undiscovered and software inventory is incomplete.	- Automatic generation at release - CI/CD integration - Documented treatment of known unknowns and missing metadata
Outdated SBOM	The SBOM no longer reflects the current version of the software.	Insufficient vulnerability management and inaccurate risk assessment.	- Regenerate SBOMs based on the above identified triggers
Poor component identification	Components are not uniquely identified across ecosystems.	Inaccurate vulnerability correlation and matching.	Use standardised ecosystem-aware component identifiers. - Use repository URL from upstream source repository, OR - Use Package URL combined with namespace, exact version, and cryptographic hashes where applicable OR - Use CPE, following naming convention consistently with namespace, exact version and cryptographic hashes where applicable
Insufficient SBOM Accuracy	SBOM generation tools relying on heuristic analysis to identify components and dependencies, generate SBOMs that may contain components that are not actually present or used (false positives), omit components that are present (false negatives), or incorrectly report versions, dependency relationships, or identifiers.	Excessive vulnerability alerts and alert fatigue due to false-positive components, missed vulnerabilities due to omitted or incorrectly identified components and/ or incorrect vulnerability matching.	- Select appropriate SBOM generation techniques according to the software ecosystem, build process, artifact type, and intended SBOM use case, - Validate generated SBOMs against S/W artifacts, - Perform completeness and accuracy checks, - Validate component identifiers
Weak SBOM-to-Artifact Binding	SBOMs not strongly and unambiguously associated with the exact software artifact, build, release, container image, or package that it describes.	Incorrect SBOM may be used for vulnerability analysis increasing false positives or false negatives in vulnerability matching.	- Cryptographically bind the SBOM to the exact artifact using a cryptographic hash or immutable digest - Verify the SBOM-to-artifact association

Limited SBOM operationalization	SBOMs are generated but not integrated into the SDLC.	Reduced value from SBOM adoption and ineffective vulnerability management.	- Consume SBOMs for asset management, vulnerability management, license compliance etc.
Insufficient SBOM Retention and Lifecycle Evidence	Failure to retain historical SBOMs and their associated evidence (e.g. previous SBOM versions, build identifiers, artifact digests, signatures, VEX documents) for an adequate period.	Inability to determine whether historical or deployed software versions contain a newly discovered vulnerable component.	- Define an SBOM retention policy - Retain SBOMs per release - Maintain version history - Maintain SBOM-to-artifact traceability - Define lifecycle disposal procedures
Lack of vulnerability prioritization	Every vulnerability identified through the SBOM is treated equally without considering exploitability.	Vulnerability flooding and delayed remediation of highly targeted vulnerabilities.	- Combine SBOM data with VEX to assess exploitability and reachability - Incorporate asset criticality and exposure - Establish risk-based vulnerability prioritization process

Table 6 - SBOM General Risks

4. SBOM Best Practices

This section presents indicative best practices for secure SBOM integration and automation based on the defined SBOM lifecycle while also complementing the suggested mitigation activities for the identified threats and general risks presented in the previous section.

4.1 SBOM Integration and Automation

For a minimum viable implementation, begin with build-time integration as the foundation. This approach captures dependencies when they are most accurately known and requires minimal infrastructure changes.

Leverage existing continuous integration infrastructure by adding SBOM generation as a pipeline stage. Utilise command-line interfaces compatible with current build environments and store generated SBOMs alongside build artifacts using existing artifact repositories. Finally, implement validation through standard pipeline testing mechanisms. (Figure 3)



Figure 3 - Pipeline Integration

4.1.1 Integration Patterns

Build-Time Integration Pattern

Integrate generation immediately following successful compilation. Capture both direct and transitive dependencies while build context remains available. Generate supplementary metadata including build environment details and compilation parameters. Many build

environments such as Yocto, Kubernetes, Linux, Zephyr, etc. have automatic SBOM generation capabilities already integrated, and the build just needs to enable them.

Event-Triggered Generation Pattern

Configure repository webhooks to initiate generation upon specific events. Support multiple trigger types including commit or pull request, and release events. Implement queuing mechanisms to handle concurrent generation requests. Maintain generation history for audit and rollback capabilities.

4.1.2 Common Configuration Requirements

Repository Structure Organisation

Establish a dedicated directory for SBOM storage within the repository structure. Maintain separation between source SBOMs and generated outputs. Implement naming conventions correlating SBOMs with their associated components and configure ignore patterns preventing SBOM inclusion in builds.

Pipeline Orchestration Templates

Define reusable pipeline segments for SBOM operations. Parameterise templates supporting multiple project types. Include standard validation and implement error handling and retry logic to manage transient failures.

Validation Checkpoints

Enforce format validation before artifact publication. Verify licence compatibility against organisational policies. Validate cryptographic signatures when attestation is required.

Vulnerability Management

Check for known vulnerabilities using configured vulnerability databases. Vulnerabilities identified through SBOM consumption should be linked to defined remediation timelines, severity thresholds and escalation paths. The remediation timelines should consider severity, exploitability, exposure, product criticality, availability of patches and compensating controls.

Automation Benefits

Automation eliminates manual generation inconsistencies whilst ensuring every build produces corresponding SBOM documentation. This continuous generation enables immediate vulnerability assessment when new threats emerge, while historical SBOM retention supports forensic analysis and compliance demonstration. Finally, automated validation prevents non-compliant SBOMs from entering production environments.

Prerequisites for Success

Successful integration requires mature continuous integration infrastructure and standardised build processes. Clear policies regarding generation triggers, storage locations, and retention

periods must be established, while monitoring capabilities should track generation success rates and identify systematic failures.

SBOM processes should generate evidence for internal audits, conformity assessments, incident response and supervisory requests. This evidence includes generation logs, validation results, signatures, timestamps, release links and retention records.

Finally, organisations should document known gaps, exceptions and “known unknowns” in SBOM coverage, especially where some dependencies cannot be identified, supplier information is unavailable, or binary-only components cannot be fully analysed.

4.2 General Security Recommendations

Security controls should be applied end-to-end across the SBOM CI/CD pipeline to protect the main principles of confidentiality, integrity, availability, authenticity and non-repudiation. Table 7 presents indicative security controls to be taken into consideration when building the CI/CD pipelines and their purpose.

Lifecycle Phase	Security control	Purpose
All Phases	Role-based Access Control (RBAC)	Restrict access to repositories, CI/CD workflows, signing service and SBOM repositories based on least privilege.
	Multi-factor Authentication (MFA)	Protect developer, administrator and CI/CD platform accounts from credential compromise.
	Least Privilege	Minimize permissions to users, service accounts and automation.
	Secure Transport Layer	Protect communications between pipeline components using TLS 1.3 with certificate validation.
	Audit Logging	Record all pipeline activities, including generation, validation, signing events etc.
	Security Event Alerting	Generate alerts for failed validations, signature failures and unauthorized access.
	Pipeline Integrity Protection	Protect CI/CD workflows from unauthorized modification through code review and branch protection.
	Configuration Management	Maintain version-controlled and approved configurations for pipeline components and security policies.
Generation	Tool stack verification	Verify the integrity and provenance of SBOM generators, validators and supporting tools before execution.
	Trusted repositories	Retrieve dependencies only from authenticated and trusted registries or mirrors.

	Provenance Generation	Generate verifiable build provenance linking source, build process, SBOM and software artifact.
Validation, Signing, Store	Secure Secrets Management	Store signing keys, API tokens and repository credentials in secure vaults where applicable.
Signing, Store	Artifact Binding	Cryptographically associate the SBOM with the software artifact it describes.
Store	Tamper-Evident Storage	Store SBOMs and artifacts in immutable or version-controlled repositories with integrity protection.
	Repository Integrity Monitoring	Detect unauthorized changes to source code, artifacts or SBOM repositories.
Store, Maintenance & Update	Version control	Maintain version history of software artifacts and associated SBOMs.
	Backup and Recovery	Ensure SBOM repositories, signing material and audit logs can be recovered.

Table 7 - Security Recommendations

5. Bibliography

REGULATION (EU) 2024/2847 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL, 2024. Horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) No 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act).

European Union Agency for Cybersecurity (ENISA), 2026. *SBOM Adoption State of Play - 2026*.

European Union Agency for Cybersecurity (ENISA), 2026. *Secure by Design and Secure by Default playbook Version 1.1*.

European Union Agency for Cybersecurity (ENISA), 2026. *Technical Advisory for Secure Use of Package Managers Version 1.1*.

German Federal Office for Information Security (BSI), 2025. *BSI TR-03183-2: Cyber Resilience Requirements for Manufacturers and Products - Part2: Software Bill of Materials (SBOM) Version 2.1.0*.

U.S. Cybersecurity and Infrastructure Security Agency (CISA), 2026. *Minimum Elements for a Software Bill of Materials (SBOM)*.

Open Worldwide Application Security Project (OWASP), 2025. *Authoritative Guide to SBOM - Implement and optimize use of Software Bill of Materials Third Edition*.

The Linux Foundation and its Contributors, 2024. *The System Package Data Exchange (SPDX®) Specification Version 3.0.1*.

U.S. Cybersecurity and Infrastructure Security Agency (CISA) and SBOM Tooling and Implementation WG Contributors, 2023. *Types of Software Bill of Materials*.

U.S. Cybersecurity and Infrastructure Security Agency (CISA) and SBOM Tooling and Implementation WG Contributors, 2024. *Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) Third Edition*.

SBOM Sharing & Exchanging Working Group, 2024. *SBOM Sharing Roles and Considerations*

Open Source Security Foundation (OpenSSF) Best Practices Working Group, 2023. *Source Code Management Platform Configuration Best Practices*.

Open Source Security Foundation (OpenSSF) Best Practices Working Group, 2023. *Concise Guide for Developing More Secure Software*.

Open Source Security Foundation (OpenSSF) and its Contributors, 2025. *Supply-chain Levels for Software Artifacts Version 1.2*.

Open Source Security Foundation (OpenSSF), 2025. *Improving Risk Management Decisions with SBOM Data*.

Open Source Security Foundation (OpenSSF) and its Contributors, 2025. Supply-chain Levels for Software Artifacts (SLSA) Version 1.2.

Cloud Native Computing Foundation (CNCF), 2026. *in-toto Attestation Framework Spec Version 1.2*.

Open Worldwide Application Security Project (OWASP), 2020. *Software Component Verification Standard Version 1.0*.

Open Worldwide Application Security Project (OWASP), 2025. Software Supply Chain Security Cheat Sheet.

Open Worldwide Application Security Project (OWASP), 2025. Dependency Graph & SBOM Best Practices Cheat Sheet

National Telecommunications and Information Administration (NTIA), 2021. *How-To Guide for SBOM Generation*.

National Telecommunications and Information Administration (NTIA), 2021. *Software Suppliers Playbook: SBOM Production and Provision Version 1.0*.

Microsoft. *The STRIDE Threat Model*.

Open Source Security Foundation (OpenSSF) End Users Working Group, 2024. *Threat Model of Enterprise Open Source Supply Chains*.

U.S. National Institute of Standards and Technology (NIST), 2022. *NIST Special Publication 800-218 - Secure Software Development Framework (SSDF) Version 1.1*.

Open Worldwide Application Security Project (OWASP), 2026. *Dependency Track Documentation Version 4.14*.

Sigstore Project. *Cosign Documentation*.

Open Source Security Foundation (OpenSSF) SBOM Everywhere WG, 2025. *SBOM Landscape*.

ABOUT ENISA

The European Union Agency for Cybersecurity, ENISA, is the Union's agency dedicated to achieving a high common level of cybersecurity across Europe. Established in 2004 and strengthened by the EU Cybersecurity Act, the European Union Agency for Cybersecurity contributes to EU cyber policy, enhances the trustworthiness of ICT products, services and processes with cybersecurity certification schemes, cooperates with Member States and EU bodies, and helps Europe prepare for the cyber challenges of tomorrow. Through knowledge sharing, capacity building and awareness raising, the Agency works together with its key stakeholders to strengthen trust in the connected economy, to boost resilience of the Union's infrastructure, and, ultimately, to keep Europe's society and citizens digitally secure. More information about ENISA and its work can be found here: www.enisa.europa.eu.

ENISA

European Union Agency for Cybersecurity

Athens Office

Agamemnonos 14
Chalandri 15231, Attiki, Greece

Brussels Office

Rue de la Loi 107
1049 Brussels, Belgium



Publications Office
of the European Union

