certin
Enhancing Cyber Security in India

सत्यमेव जयते

Technical Guidelines on
| SBOM | QBOM & CBOM | AIBOM | HBOM |

Version 2.0

Indian Computer Emergency Response Team (CERT-In)
Ministry of Electronics and Information Technology
Government of India

Version 2.0 Dated 09.07.2025

Table of Contents

Tables & Figures List

1. Executive Summary

Software products are composed of many different components, some of which might come from third party sources. These third-party components and dependencies can have vulnerabilities, which attackers can exploit, leading to security incident or breaches. Key threats include attackers inserting malicious code, vulnerabilities in outdated components, and breaches by compromised suppliers. These issues can lead to data breaches, operational disruptions, and reputational damage.

These threats can be countered by maintaining visibility & transparency on software components used for building or developing the software. Software Bill of Materials (SBOM) helps organizations know exactly what components are in their software or assets, making it easier to identify and fix vulnerabilities. By using SBOMs, entities can improve their software security and protect against potential threats.

A **Software Bill of Materials (SBOM)** is list of all the components, libraries, and modules that make up a software, providing transparency into its composition. Software composition is important to comprehend as it grows more sophisticated and depends on more external components. In cybersecurity, safeguarding software against cyberattacks requires an awareness of the dependencies and components utilized in its construction. An SBOM is therefore a crucial instrument in contemporary cybersecurity procedures.

An SBOM is vital for maintaining software security. It helps organizations understand what their software is made of, manage potential risks, respond to security issues, and comply with regulations. Following are the key benefits an organization can derive by implementing SBOM:

i. **Enhanced Security Management:** By knowing the components of the software, organizations can identify which components might be vulnerable to security threats for mitigation.
ii. **Effective Incident Response:** In the event of a cyber-security incident, an SBOM assists in speeding up incident response by providing detailed component information.
iii. **Vulnerabilities Identification and Patch Management:** By listing all components, organizations can quickly spot and address known vulnerabilities in the software by patching them.

iv. **Supply Chain Security:** Supply chain risks can be reduced significantly by gaining visibility into third-party components used in creating a software.

v. **Assist in Ensuring Compliance:** SBOM helps organizations to streamline adherence to security regulations, guidelines and best practices on software security by providing required transparency in software composition.

vi. **Improved Operational Efficiency:** With a clear understanding of software components, organizations can streamline their vulnerability management processes, saving time and resources.

Indian Computer Emergency Response Team (CERT-In) has released following technical SBOM guidelines for entities, particularly those in the public sector, government, essential services, organizations involved in software export and software services industry.

Departments and organizations are encouraged to make the creation and provision of Software Bill of Materials (SBOM) a mandatory standard practice as part of software procurement and software development in order to enhance security and reduce the risk of cyber threats.

The following chapters delve into various technical aspects of the Software Bill of Materials (SBOM) explaining its purpose, and its growing significance in the software supply chain ecosystem. Second chapter provides overview of SBOM and discuss about the scope and implementation of SBOM, followed by a chapter on SBOM ecosystem, which explains different levels and classifications of SBOM. Subsequent chapters, explore the different standards and data formats employed for representing SBOM information, and elaborate on the minimum elements, data fields, and automation support. Objectives of all processes and practices involved in SBOM, secure SBOM sharing, and distribution are elaborated in this document including approaches for vulnerability tracking and analysis in SBOM. Finally, the last chapter of the document covers recommendations and best practices for SBOM implementation.

2. Overview of SBOM

2.1 Necessity and Utilization

Increasing software complexity emphasizes the necessity of SBOM, serving as the foundation for Software Composition Analysis (SCA) tools, aiding in vulnerability detection, license compliance and instrumental in vendor risk management. The production of software-defined systems has considerably expanded the cyber threat landscape, with adversaries increasingly targeting the software supply chain to infiltrate sensitive systems and data.

In order to improve security, compliance, risk management, supply chain transparency, quality assurance, interoperability, and vendor management in their software development and procurement processes, departments/ organizations are encouraged to prioritize the creation and provision of Software Bill of Materials (SBOM) as a standard practice. Organizations should thoroughly analyze the critical components involved in any stage of the software lifecycle - including design, development, analysis, deployment, maintenance, and update - and mandate SBOM usage. SBOM help serve three main purposes, as follows:

- Implementing SBOM can assist government departments and organizations in making informed pre-acquisition decisions for software purchases.
- Adopting SBOM can facilitate vulnerability management, asset tracking and compliance across the government entities and essential sector organisations.
- SBOM implementation can aid organizations in Software development and maintenance of their products.

It is recommended that all the Government, Public Sector and Essential Services Organizations should include requirements for SBOM in all their software and solutions purchase/procurement. It is also recommended that security teams of user organizations should include SBOM inventory in work flow of vulnerability management.

2.2 Application & Scope

This guideline has been issued by Indian Computer Emergency Response Team (CERT-In) for the following entities especially in the Government, Public Sector, Essential Services Organizations and organizations involved with software exports and software services industry:

i. Software Consumer - Organizations that acquire software applications to support their operations, enhance productivity, and achieve their business objectives.
ii. Software Developer- Organizations that develop customized software solutions.
iii. System Integrator/Software Reseller- Organizations that distribute the software products and also provide value-added services including customization, integration, support, and training.

SBOM are becoming an essential tool for visibility, vulnerability patching, reducing exposure and quick response. For instance, a typical organization relies on a vast network of interconnected systems, end points, control systems, automation software, and operational technology (OT) components. Maintaining accurate SBOM for these complex IT and OT environments allows security teams to better understand their attack surface and respond more effectively to vulnerabilities. This proactive approach helps organizations safeguard their operations and ensure resilience against cyber threats.

For example, in financial institutions SBOM proves invaluable from a cybersecurity standpoint. Banks and fintech companies often utilize a wide array of commercial off-the-shelf (COTS) software, open-source libraries, and custom-developed components to power their digital services and backend systems. Maintaining an up-to-date SBOM for this heterogeneous software landscape enables security teams to rapidly identify and mitigate vulnerabilities, comply with industry regulations, and better manage supply chain risks.

The use-cases of the SBOM, with respect to software development, supply chain management, cybersecurity, and regulatory compliance are:

2.2.1 Software Development and Maintenance: SBOM provides a detailed inventory of the components and dependencies that make up a software system. This information allows developers to more effectively manage vulnerabilities, track licenses, and monitor the provenance of their software. Maintaining an accurate and up-to-date SBOM is crucial for organizations to understand their software supply chain risks and take proactive measures so as to ensure the security and integrity of their applications.

2.2.2 Supply Chain Management: SBOM provides transparency into the software supply chain, allowing organizations to assess the security and reliability of third-party components. It helps in identifying potential risks associated with the use of third-party libraries or components and facilitates informed decision-making about procurement and vendor management.

2.2.3 **Cybersecurity**: SBOM helps in integration with existing security tools, automating vulnerability detection, remediation and plays a crucial role in cybersecurity practices. SBOM provides visibility into the software components and their dependencies, enabling organizations to identify and mitigate security vulnerabilities effectively. By having a comprehensive understanding of software composition, organizations can quickly respond to security incidents, patch vulnerabilities, and ensure the integrity and security of their software systems.

2.2.4 **Regulatory Compliance**: SBOM is increasingly becoming a requirement for regulatory compliance in various industries, especially in sectors dealing with essential services such as healthcare, finance, and government. Globally regulators are recognizing the SBOM as a promising tool and are emphasizing SBOM adoptions through their regulations such as EU Cyber Resilience Act.

2.2.5 **Risk Management**: SBOM supports risk management efforts by providing insights into the software supply chain. Organizations can assess the potential risks associated with specific software components, such as known vulnerabilities, license conflicts, or deprecated libraries. By proactively managing these risks, organizations can enhance the resilience of their software systems and minimize the likelihood of security breaches or compliance issues.

2.2.6 **Interoperability and Compatibility**: SBOM facilitates interoperability and compatibility testing by providing detailed information about software components and their versions. This helps in ensuring that different software systems can work together seamlessly without compatibility issues, and hence improving the overall quality and reliability of software products.

2.3 SBOM Implementation

SBOM should be implemented for every new software component release and updated promptly for any changes such as updates, upgrades, releases, and patches. The accuracy of SBOM is maintained by updating whenever there is a new information about included components, regardless of whether the components themselves have changed. When modifying existing components, choose a consistent approach: either treat the change as a new component or update the existing one. For clarity, use standardized versioning methods throughout.

Consider the following scenario in an organization:

A. Government organization ***GovInsights*** hires a software development company ***TechGenius*** to develop a data analytics application ***DataAnalyzer***

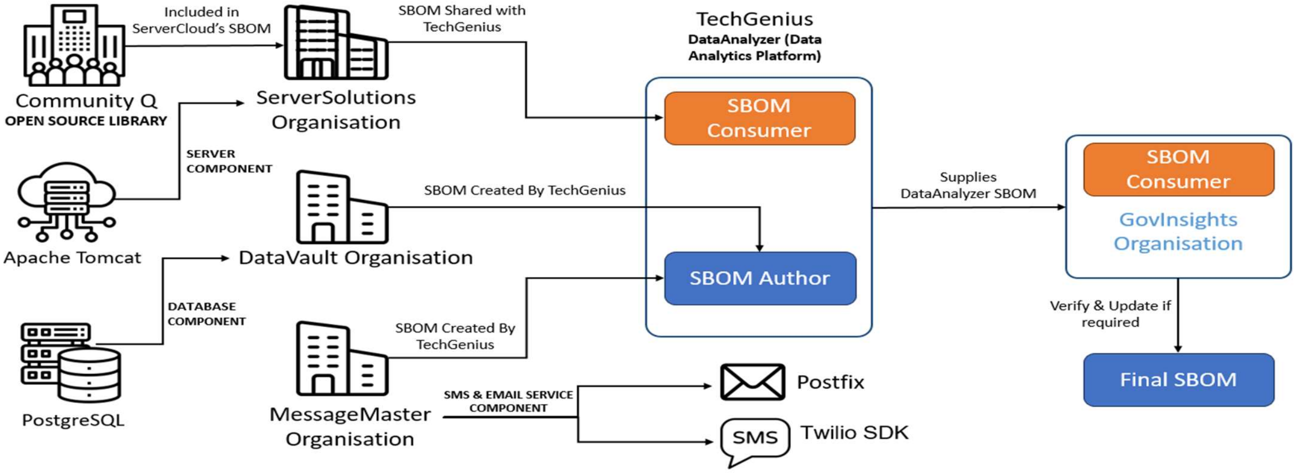
B. In order to develop the product: ***DataAnalyzer***, Company: ***TechGenius*** uses the following components:
   1. SMS and Email services namely **Postfix & Twilio SDK** by the company: ***MessageMaster.***
   2. Database component: ***PostgreSQL*** by the company: ***DataVault***
   3. ***Apache Tomcat Server*** provided by company: ***ServerSolutions*** which has used many open source libraries for their server.

Various components/software in the aforementioned scenario and their corresponding SBOM type and the status is provided in the table below:

Table 1: Software Components and the SBOM Author Status in the aforementioned scenario

S. No	Name	SBOM Author Status
1	SBOM for ***DataAnalyzer*** application	To be developed by Company ***TechGenius*** and will be provided to ***GovInsights*** organisation along with the product/application i.e. ***DataAnalyzer***
2	SBOM for ***PostgreSQL***	Top level SBOM was developed by **TechGenius** as **DataVault** never created a SBOM for this component
3	SBOM for the platform ***Apache Tomcat Server***	Delivery SBOM was created by the company ***ServerSolutions*** and shared with **TechGenius** when the platform was procured by ***TechGenius***
4	SBOM for the **Postfix & Twilio SDK**	Transitive SBOM was created by the company **TechGenius** as SBOM was not made available by ***MessageMaster***

The interrelationships among the stakeholders and components in this scenario are visually represented in Figure 1. As depicted, numerous entities within the SBOM ecosystem, function as both providers and consumers of software. This entails not only utilizing information from an SBOM provided by another entity but also participating in the creation of an SBOM for newly developed components and subsequently sharing it with other entities. Ideally, the creator of a software component should also be responsible for authoring the corresponding SBOM.


Community Q
OPEN SOURCE LIBRARY
Included in ServerCloud's SBOM
ServerSolutions Organisation
SBOM Shared with TechGenius
TechGenius
DataAnalyzer (Data Analytics Platform)
SBOM Consumer
SBOM Author
Apache Tomcat
SERVER COMPONENT
DataVault Organisation
SBOM Created By TechGenius
PostgreSQL
DATABASE COMPONENT
MessageMaster Organisation
SBOM Created By TechGenius
SMS & EMAIL SERVICE COMPONENT
Postfix
SMS
Twilio SDK
Supplies DataAnalyzer SBOM
SBOM Consumer
GovInsights Organisation
Verify & Update if required
Final SBOM

- SBOM Consumer: Must ask for complete SBOM.
- Software Developer: Must ensure that correct and complete SBOM is supplied to consumer.

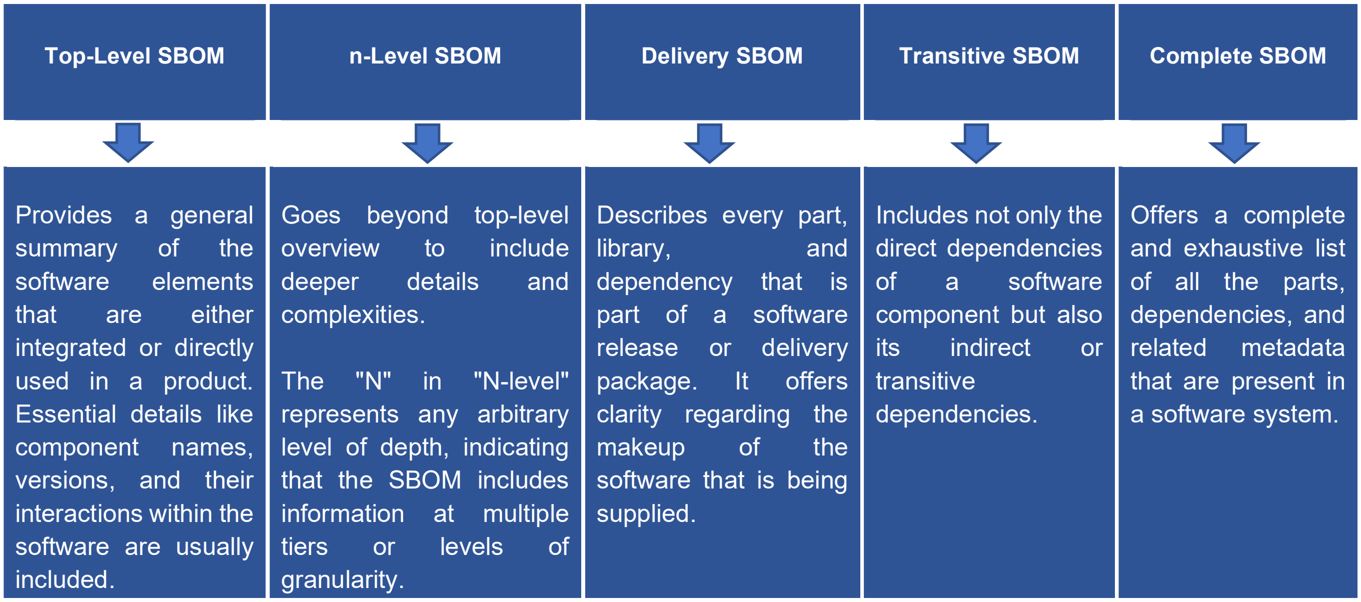
Figure 1: Flow Chart for the Scenario

3. Ecosystem

SBOM ecosystem encompasses the network of stakeholders, tools, standards, and processes involved in the creation, distribution, analysis, and utilization of SBOM across the software supply chain. This section describes an approach for Software Consumer/Software Developer/ System Integrator organizations to develop a SBOM ecosystem at an organizational level. This section also explains different classification of SBOM.

3.1 Levels of SBOM

The different levels of SBOM, each offering varying degrees of granularity and complexity indicates specific needs and the complexity of their respective software environments. Organizations should choose to implement one or more SBOM levels to achieve the efficient balance of transparency, risk management, and operational efficiency.



Top-Level SBOM	n-Level SBOM	Delivery SBOM	Transitive SBOM	Complete SBOM
Provides a general summary of the software elements that are either integrated or directly used in a product. Essential details like component names, versions, and their interactions within the software are usually included.	Goes beyond top-level overview to include deeper details and complexities. The "N" in "N-level" represents any arbitrary level of depth, indicating that the SBOM includes information at multiple tiers or levels of granularity.	Describes every part, library, and dependency that is part of a software release or delivery package. It offers clarity regarding the makeup of the software that is being supplied.	Includes not only the direct dependencies of a software component but also its indirect or transitive dependencies.	Offers a complete and exhaustive list of all the parts, dependencies, and related metadata that are present in a software system.

Figure 2: Levels of SBOM

Adopting a multiple SBOM approach can significantly enhance an organization's cyber resilience. Organizations should create a customized SBOM for consumers, addressing security requirements without exposing sensitive data. Concurrently, they should maintain an internal SBOM at the "complete" level to identify and share vulnerability updates specific to that software in detail with the consumer periodically on a mandatory basis. This approach balances cyber resilience, data confidentiality, and collaborative security across the ecosystem, especially in scenarios where organizations face constraints or apprehensions regarding data leaks and intellectual property theft resulting from sharing complete software and dependency details.

Table 2: Mapping of level as per Scenario for organization who created SBOM

S. No	Name	SBOM level	SBOM Author Status
1	SBOM for ***DataAnalyzer*** application	Complete SBOM	To be developed by Company ***TechGenius*** and will be provided to ***GovInsights*** organization along with the product/application i.e. ***DataAnalyzer***
2	SBOM for ***PostgreSQL***	Top level SBOM	Top level SBOM was developed by **TechGenius** as **DataVault** never created a SBOM for this component
3	SBOM for the platform ***Apache Tomcat Server***	Delivery SBOM	Delivery SBOM was created by the company ***ServerSolutions*** and shared with **TechGenius** when the platform was procured by ***TechGenius***
4	SBOM for the **Postfix & Twilio SDK**	Transitive SBOM	Transitive SBOM was created by the company **TechGenius** as SBOM was not made available by ***MessageMaster***

3.2 Classification of SBOM

SBOM classifications align with stages in the software development lifecycle, each providing distinct data and insights. Different classifications of SBOM have been depicted in Figure 3.

3.2.1 The Design SBOM captures planned components, even before they exist.

3.2.2 The Source SBOM reflects the development environment, including source files and dependencies.

3.2.3 The Build SBOM is generated during the build process, incorporating details like source files, dependencies, and pre-built components.

3.2.4 The Analyzed SBOM is created by inspecting final software artifacts post-build.

3.2.5 The Deployed SBOM provides an inventory of the software installed and configured on a specific system, combining information from various SBOM types and taking into account the deployment environment.

3.2.6 The Runtime SBOM is created by monitoring active software components, including their external interactions and dynamically loaded dependencies, during runtime execution.

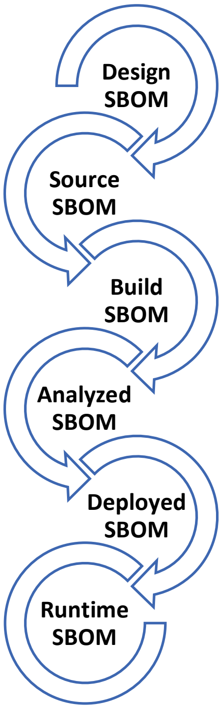
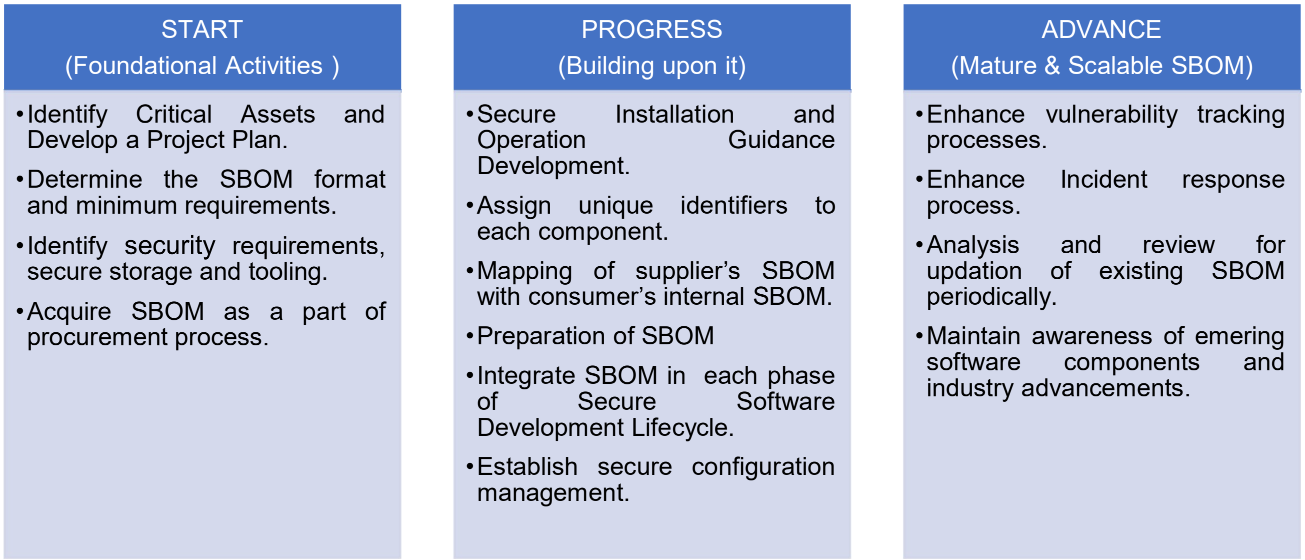


Figure 3: SBOM Classification Aligned with SDLC Stages

3.3 Roadmap for Organizations to develop and adopt SBOM

To establish a SBOM ecosystem within an organization the development of an SBOM program should follow a phased approach, starting from a basic foundation (START), then building upon it (PROGRESS), and ultimately reaching a mature and scalable SBOM implementation (ADVANCE). The order of activities is indicative. Organization may choose to move an activity up or down depending on their overall security requirements, project timeline and resource availability.



START
(Foundational Activities)

- Identify Critical Assets and Develop a Project Plan.
- Determine the SBOM format and minimum requirements.
- Identify security requirements, secure storage and tooling.
- Acquire SBOM as a part of procurement process.

PROGRESS
(Building upon it)

- Secure Installation and Operation Guidance Development.
- Assign unique identifiers to each component.
- Mapping of supplier's SBOM with consumer's internal SBOM.
- Preparation of SBOM
- Integrate SBOM in each phase of Secure Software Development Lifecycle.
- Establish secure configuration management.

ADVANCE
(Mature & Scalable SBOM)

- Enhance vulnerability tracking processes.
- Enhance Incident response process.
- Analysis and review for updation of existing SBOM periodically.
- Maintain awareness of emering software components and industry advancements.

Figure 4: Steps & its Activities for Developing SBOM Ecosystem at Organizational Level

3.3.1 PHASE-1 (START): The foundational activities will lay the groundwork for the SBOM program. It is likely the first SBOM will be acquired from suppliers during the procurement process. Since the software can vary in terms of its architecture, existing resources, budget, availability of qualified individuals, the intent of this phase is to establish methods that enable kick-start of the SBOM ecosystem within an organization.

3.3.1.1 Identify Critical Assets and Develop a Project Plan: Develop a comprehensive project plan that define roles, responsibilities, timelines, and resource requirements. Alongside the project plan, identify the change management requirements to ensure stakeholder are onboard for the new SBOM processes.

3.3.1.2 Determine the SBOM format and minimum requirement: Defining the SBOM format and minimum data requirements before its creation is critical. It ensures a standardized, machine-readable structure that enables consistent sharing and processing across the supply chain.

3.3.1.3 Identify security requirements, secure storage and tooling: This entails determining the appropriate classification and handling procedures in line with site security policies. Next, organizations should establish secure storage for SBOM, initially segregating individual SBOM in dedicated repositories. As the SBOM program matures, integration with asset management applications should be pursued, along with linking to other security-related information like vulnerability data.

3.3.1.4 **Acquire SBOM as a part of procurement process:** By requiring SBOM provision by suppliers in purchase orders or contracts, specifying SBOM elements, delivery timeframe, and delivery method, transparency is ensured, facilitating the SBOM integration process.

3.3.2 PHASE – 2 (PROGRESS): This involves sustaining activities resulting in establishment of secure installation and configuration management, along with integrating unique component identification to address Supplier and Component namespace issues. Integration with Secure Software Development Life Cycle (SSDLC) by Software Developer Organization will begin to provide actionable security information to secure the software in its build phase.

3.3.2.1 **Secure Installation and Operation Guidance Development:** A comprehensive checklist for secure software installation and operation, tailored to the target consumer's technology sector and usage needs should be created by the supplier in correlation with consumer organization. To ensure secure operations, a set of key checklist pointers can be derived from the Guidelines for Secure Application Design, Development, Implementation & Operations guidelines, highlighting essential considerations that should be addressed throughout the deployment and operational phases of an application's lifecycle.

3.3.2.2 **Assign unique identifiers to each component**: Consumers may overlook security updates or vulnerabilities if unaware of rebranding, leaving them vulnerable to exploitation. This makes it challenging for consumers to research the accurate data fields, such as current supplier and component names, to include in their own SBOM. Supplier and component name changes would result in an SBOM revision and a link from the old SBOM to its successor to maintain revision history. However, in cases where the historical context may be unknown to the consumer, mapping older names to current ones can be problematic, especially if the original supplier no longer exists. To address this, a unique identifier should be created. This identifier may follow the following structure:

Table 3: Syntax for Unique Identifier and Example

Field	Description	Example
scheme	Indicates the format of the identifier, in this case, pkg for the Package URL (PURL) format.	pkg
type	Specifies the type of the identifier, in this case, supplier to represent the supplier of the software component.	Supplier
namespace	Identifies the name of the organization or entity that is the supplier of the software component.	Apache Software Foundation
name	Provides the name of the software component itself.	Apache Tomcat
version	Denotes the specific version of the software component.	9.0.71
qualifiers (optional)	Allows for the inclusion of additional contextual information about the software component, such as architecture, operating system, or other metadata.	arch=x86_64&os=linux
subpath (optional)	Can be used to specify a subpath or location within the software component, if applicable.	#server/webapps

The unique identifier for the scenario would be: pkg:supplier/ApacheSoftwareFoundation/ApacheTomcat@9.0.71?arch=x86_64&os=linux#server/webapps

Table 4: Utility of Unique Identifiers

Issue	Apache Tomcat Example	How the Unique Identifier Helps
Ownership and Branding Changes	Initially, Apache Tomcat was developed and maintained by the Apache Software Foundation. Over time, the ownership could change, and the new owner might rebrand the project (e.g., "TomcatX" or "Acme Tomcat").	The unique identifier pkg:supplier/Apache Software Foundation/Apache Tomcat@9.0.71?arch=x86_64&os=linux would still be valid, even with ownership and branding changes. The consumer can update the SBOM with the new identifier pkg:supplier/Acme Corp/TomcatX@9.0.71?arch=x86_64&os=linux,

Issue	Apache Tomcat Example	How the Unique Identifier Helps
		maintaining the linkage between old and new component names.
Version Ambiguity	The vendor releases a new version of Apache Tomcat (e.g., 10.0.0), but keeps the same component name.	The unique identifier pkg:supplier/Apache Software Foundation/Apache Tomcat@9.0.71?arch=x86_64&os=linux clearly indicates the specific version (9.0.71). When a new version is released, the consumer can update the SBOM with pkg:supplier/Apache Software Foundation/Apache Tomcat@10.0.0?arch=x86_64&os=linux, eliminating version ambiguity.

3.3.2.3 **Mapping of supplier's SBOM with consumer's internal SBOM:** The consumer organization should map and develop an internal SBOM on the basis of SBOM provided by the supplier. It should also include author name (personnel of consumer organization) and timestamp to trace the integrity and efficient updating of the developer of that internal SBOM.

3.3.2.4 **Preparation of SBOM:** SBOM should be prepared by both supplier and consumer organization. Identify installed components with vulnerabilities by correlating the known vulnerability data and vendor vulnerability attestations. Known vulnerability data is available through various sources, including vendor notifications, third-party notifications, and data repositories. On this basis, a complete-level SBOM should be generated either internally by organization or externally by the software vendor for enhancing the security and visibility of supply chain attacks.

3.3.2.5 **Integrate SBOM in each phase of Secure Software Development Lifecycle (SSDLC):** SBOM can be incorporated into each phase of the SSDLC by Software Developer organization in such a way that during design, SBOM should inform decisions regarding component selection and potential security risks. The use of SBOM during software development can improve efficiencies and provide greater insight into build and source components, as well as product functionality, for both the developer and user.

3.3.2.6 **Establish secure configuration management:** Implement stringent access controls encryption, periodic audits of software, and integration with security frameworks to ensure secure configuration management in SBOM.

3.3.3 PHASE-3 (ADVANCE): Enhancing activities related for monitoring of vulnerabilities and seamless integration of SBOM with security orchestration tools for vulnerability management and incident response.

3.3.3.1 **Enhance vulnerability tracking processes:** Capture vulnerability information associated with SBOM. Historical vulnerability information should be integrated into the SBOM ecosystem, and specialists should have procedures such as cross-referencing the identified vulnerabilities with the components listed in the SBOM repository and checking the equipment database for relevant configuration data to assess the impact and potential mitigation measures for tracking and analysis of known-vulnerabilities.

3.3.3.2 **Enhance Incident response process:** CERT-In issues alerts, vulnerabilities notes and advisories on various threats. Often, these threats are associated with newly disclosed software vulnerabilities. Organization should establish threat hunting teams that use this information to determine if their organizations are vulnerable to the newly discovered threat and whether they have been compromised by it.

3.3.3.3 **Periodic Analysis and Review for Updating Existing SBOM:** This involves checking if software components and their dependencies are as per latest records, ensuring timely updates.

3.3.3.4 **Maintain awareness of emerging software components and industry advancements:** Organizations are encouraged to uphold SBOM awareness programs either independently or in collaboration to third-party organizations to share the information on emerging SBOM formats, data elements, its implementation in organization in adherence to challenges faced by SBOM practitioners.

3.4 License Management

License management is an early use case for SBOM, helping organizations with large and complex software portfolios track the licenses and terms of their diverse software components, especially for open-source software. SBOM can convey data about the licenses for each component. This data can also allow the consumer to know if the software can be used as a component of another application without creating legal risk. License information for components included in software can be checked to prevent negligence in compliance, thus reducing the risk of license violations and the workloads required for license management. Following practices streamlines license management processes and helps mitigate risks associated with non-compliance.

a) Consumer should be able to view the licenses of all individual components within a Product being evaluated, alongside the Product's own license. This provides the user with better insight when selecting a product and determining the suitable license arrangement for their business requirements or application

b) Identify each software license using an identifier (e.g. SPDX identifier). These identifiers, along with expressions, serve as unique codes that represent specific license terms and conditions. By leveraging these identifiers, organizations should efficiently manage and understand the licensing obligations associated with their software assets.

c) An alternative license database should be considered, if the license identifiers cannot be found in the primary one, such as the Scancode LicenseDB AboutCode. These alternative identifiers should be prefixed (e.g. "LicenseRef-scancode-") to indicate their origin, thus facilitating mapping and understanding.

d) When encountering licenses that are not recognized by established lists like SPDX, organizations should assign a unique identifier. This ensures proper identification and tracking of unknown licenses within their systems.

e) When modifying licenses with placeholders or templates, it is recommended to ensure that these changes don't alter the fundamental terms of the license. Instead, they should be considered part of the original license identified by its unique identifier, like those provided by SPDX License Expressions. This helps maintain clarity and consistency in license management practices.

f) When dealing with multiple licenses for software, it is important to use operators (e.g. SPDX operators) to combine them correctly. These operators help link different license identifiers together, ensuring clarity and consistency in license expressions. This ensures that the resulting license expressions accurately represent the licensing terms applicable to the software.

g) When managing licenses, any exception clauses attached to a license text should be linked to the main license identifier using appropriate operators such as "WITH” for SPDX operators. Additionally, the exception clause names should be described with identifiers following the established requirements for license identification.

h) When making slight changes to a license text, if the modifications do not significantly alter the meaning of the original license, it is recommended to use the same identifier as the original license.

4. SBOM Preparation

4.1 Minimum Elements of SBOM

Minimum Elements of the SBOM· dictates the "Data Fields" as the necessary information related to a component in a software to be considered With "Automation Support" detection and management can be enhanced by integrating with security orchestration tools and the "Process and Practice" for implementation of the SBOM in the organization. The "Minimum Elements" categories and definitions are as follows.

Table 5: Minimum Elements of SBOM

Minimum Elements	Overview	Definition
Data Fields	Document baseline information about each component that should be tracked.	This baseline component information includes: • Component Name • Component Version • Component Description • Component Supplier • Component License • Component Origin • Component Dependencies • Vulnerabilities • Patch Status • Release Date • End-of-Life (EOL) Date • Criticality • Usage Restrictions • Checksums or Hashes • Comments or Notes • Author of SBOM Data • Timestamp • Executable Property • Archive Property

Minimum Elements	Overview	Definition
		• Structured Property • Unique Identifier
Automation Support	Support automation, including via automatic generation and machine-readability to allow for scaling across the software ecosystem.	Data formats used to generate and consume SBOM include • Software Package Data Exchange (SPDX) • CycloneDX
Practices and Processes	Define the operations of SBOM requests, generation and use.	Organizations definition of SBOM operation procedure should be based on: • Frequency • Depth • Known Unknowns • Distribution and Delivery • Access Control • Accommodation of Mistakes

4.2 Data Fields

Data fields contain a baseline information regarding each component that needs to be tracked and maintained. The organizations can create a comprehensive inventory of software components, dependencies, and associated metadata, enabling better transparency, security, and risk management throughout the software development lifecycle.

Enabling adequate identification of these components is the aim of data fields, as it facilitates their tracking throughout the software supply chain and allows them to be mapped to other useful data sources like vulnerability or license database. The baseline components information include:

1. **Component Name**: The name of the software component or library included in the SBOM.
2. **Component Version**: The version number or identifier of the software component.

3. **Component Description**: A brief description or summary of the functionality and purpose of the software component.
4. **Component Supplier**: The entity or organization that supplied the software component, such as a vendor, third-party supplier, or open-source project.
5. **Component License**: The license under which the software component is distributed, including details such as the license type, terms, and restrictions.
6. **Component Origin**: The source or origin of the software component, such as whether it is proprietary, open-source, or obtained from a third-party vendor.
7. **Component Dependencies**: Any other software components or libraries that the current component depends on, including their names and versions.
8. **Vulnerabilities**: Information about known security vulnerabilities or weaknesses associated with the software component, including severity ratings and references to security advisories or CVE identifiers.
9. **Patch Status**: The patch or update status of the software component, indicating whether any patches or updates are available to address known vulnerabilities or issues.
10. **Release Date**: The date when the software component was released or made available for use.
11. **End-of-Life (EOL) Date**: The date when support or maintenance for the software component is scheduled to end, indicating the end of its lifecycle.
12. **Criticality**: The criticality or importance of the software component to the overall functionality or security of the application, often categorized as critical, high, medium, or low.
13. **Usage Restrictions**: Any usage restrictions or limitations associated with the software component, such as export control restrictions or intellectual property rights.
14. **Checksums or Hashes**: Cryptographic checksums or hashes of the software component files to ensure integrity and authenticity.
15. **Comments or Notes**: Additional comments, notes, or annotations relevant to the software component or its inclusion in the SBOM.
16. **Author of SBOM Data**: The name of the entity that creates the SBOM data for this component.
17. **Timestamp**: Record of the date and time of the SBOM data assembly.

18. **Executable Property**: Attributes indicating whether a component within an SBOM can be executed.
19. **Archive Property**: Characteristics denoting if a component within an SBOM is stored as an archive or compressed file.
20. **Structured Property**: Descriptors defining the organized format of data within a component listed in an SBOM.
21. **Unique Identifier**: A unique identifier is a distinct code assigned to each software component, structured as "pkg:supplier/OrganizationName/ComponentName@Version?qualifiers&subpath," aiding in tracking ownership changes and version updates, thus ensuring accurate and consistent software component management.

Table 6: Data Fields for the Components Utilised in Scenario by Organization

Component Name	Apache Tomcat	PostgreSQL	Postfix	Twilio SDK
Version	9.0.41	13.3	3.5.6	7.17.0
Description	Open-source Java web server	Open-source relational database management system	Open-source mail transfer agent (MTA)	Twilio API SDK for sending and receiving SMS
Supplier	Apache Software Foundation	PostgreSQL Global Development Group	Postfix Development Team	Twilio Inc.
License	Apache Software Foundation	PostgreSQL License	IBM Public License v1.0	Apache License 2.0
Origin	Apache License 2.0	Open-source community	Open-source community	Vendor
Dependencies	Open-source community	None	None	None
Vulnerabilities	Java Runtime Environment (JRE)	None reported	None reported	None reported
Patch Status	None reported	Up to date	Up to date	Up to date
Release Date	Up to date	May 7, 2021	October 15, 2020	January 10, 2022

Component Name	Apache Tomcat	PostgreSQL	Postfix	Twilio SDK
End of Life Date	March 22, 2021	May 7, 2026	October 15, 2025	January 10, 2027
Criticality	March 22, 2025	High	High	Medium
Usage Restrictions	High	None	None	Requires Twilio account for API access
Checksums	None	SHA-256: d7f5a6b198e75c1f438d0fa158a9bc92	SHA-256: 3bd5a7f02a81022a47a7e6cb9cb5e2b8	SHA-256: 9f3b2e5ab24a5e68a3bda6a12c1febd1
Hashes	SHA-256: 7f87a8b8ed5c23546789b8d7586219a1	MD5: b8c78139eef440fb3cb074e199b1e923	MD5: e57cb8d0ae875fda9d60291f10689e4b	MD5: 6a8c4db98ce5f0c3a92416727bc80a5e
Comments	MD5: 8937d8b1a947f45d79e457b91c2e6543	Supports SQL queries and ACID transactions.	Facilitates the delivery of emails between mail servers.	Integrates SMS functionality into applications via Twilio's cloud communications platform.
Executable Property	Yes - Contains executable binaries like catalina.sh and startup.bat.	No - Binaries like postgres are not directly executable.	No - Binaries like postfix are not directly executable.	No - The SDK itself is not directly executable, but contains libraries and modules that can be used by applications.

Component Name	Apache Tomcat	PostgreSQL	Postfix	Twilio SDK
Archive Property	No - Distributed as a directory structure.	No - Distributed as a set of installation files including postgresql.conf.	No - Distributed as a set of installation files including main.cf.	Yes - Distributed as a package or library archive file, such as twilio-python.tar.gz or twilio-java.jar.
Structured Property	Yes - Configuration files such as server.xml have defined elements.	Yes - Database schemas and files like schema.sql have structured formats.	Yes - Configuration files such as main.cf and master.cf have structured formats.	Yes - The SDK includes structured files defining API methods and configurations, such as twilio.py or twilio.xml.
Unique Identifier	pkg:supplier/ApacheSoftwareFoundation/ApacheTomcat@9.0.71?arch=x86_64&os=linux#server/webapps	pkg:supplier/PostgreSQLGlobalDevelopmentGroup/PostgreSQL@13.5?arch=x86_64&os=linux	pkg:supplier/PostfixFoundation/Postfix@3.6.2?arch=x86_64&os=linux	supplier/TwilioInc/TwilioSDK@1.20.0?arch=x86_64&os=linux

4.3 Automation Support

Supporting automation, such as automatic generation and machine-readability, enables scaling across software ecosystems and organizational boundaries. It allows for seamless integration of SBOM data into various tools and processes, facilitating collaboration and visibility across the software supply chain

Component Discovery	Version Tracking	Dependency Analysis	Vulnerability Assessment	License Compliance
Automated tools can scan software packages, repositories, and source code to identify and catalogue software components automatically. This helps in creating an initial inventory of components without manual intervention.	Automation tools can monitor software repositories and package managers to track changes and updates to software components. This ensures that SBOM remain up-to-date with the latest versions of components, reducing the risk of using outdated or vulnerable software.	Automated dependency analysis tools can identify and document dependencies between software components automatically. This helps in understanding the complex relationships between components and assessing the potential impact of changes or vulnerabilities.	Automated vulnerability scanning tools can analyze software components against known vulnerability databases, such as the National Vulnerability Database (NVD) or Common Vulnerabilities and Exposures (CVE).	Automated license scanning tools can analyze software components to identify the licenses under which they are distributed. This helps in ensuring compliance with licensing requirements and avoiding legal issues associated with the unauthorized use of proprietary or open-source software.
SBOM Generation	**Integration with DevOps Pipelines**	**Reporting and Visualization**	**Integration with Security Orchestration Platforms**	**Monitoring and Maintenance**
Automated SBOM generation tools can aggregate information from various sources, such as software repositories, package manifests, and vulnerability databases, to create comprehensive SBOM automatically. This streamlines the process of SBOM creation and ensures consistency and accuracy across multiple projects.	Automation tools can integrate SBOM generation and analysis into DevOps pipelines, allowing for continuous monitoring and assessment of software components throughout the development lifecycle. This enables proactive identification and mitigation of security risks and compliance issues.	Automated reporting and visualisation tools can generate actionable insights from SBOM data, such as identifying high-risk components, tracking compliance status, and visualising dependency graphs. This helps stakeholders make informed decisions and prioritize efforts for risk mitigation and remediation.	Automation tools can integrate with security orchestration platforms to automate remediation workflows based on SBOM analysis results. This enables automatic deployment of patches, updates, or configuration changes to mitigate security vulnerabilities quickly.	Automation tools can facilitate continuous monitoring and maintenance of SBOM by automatically updating component information, tracking changes, and generating alerts for anomalies or compliance violations.

Figure 5: Benefits of Automation Support in SBOM

Utilizing SBOM data will require tooling, which calls for consistent data formats and implementation. Automation can support various aspects of SBOM creation, maintenance, and utilization. Organizations may include this feature in their current vulnerability management procedures and audit compliance with security policies in real time. Both will depend heavily on automation, which calls for standard, machine-readable data formats. The standard format used to generate and consume SBOM is:

1. Software Package Data eXchange (SPDX)
2. CycloneDX

5. Process and Practices of SBOM for Software Consumer/Developer/Integrator Organizations

This section discusses how practitioners should perceive SBOM and what processes need to be established to address it in practice. The topics mentioned in this chapter are derived from the analysis of SBOM practices from SBOM generation, distribution and sharing, validation and verification, and vulnerability and exploitability management.

5.1 Establish Roles and Responsibilities

To implement the SBOM, identify the necessary roles and responsibilities. This should include a management sponsor, project lead, systems engineer, design engineer, procurement specialist, and operations representative. Involve additional support, such as IT, cybersecurity, and maintenance personnel, based on the project timeline and security requirements. Ensure clear ownership and collaboration across these roles to drive the SBOM implementation and integration with existing processes.

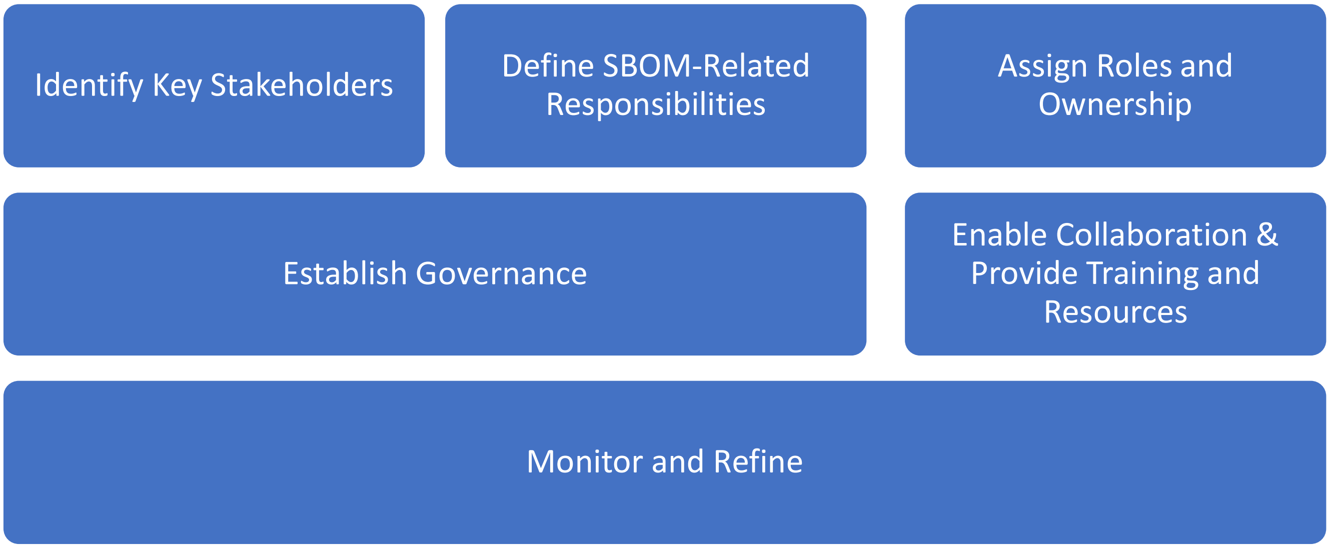

Identify Key Stakeholders
Define SBOM-Related Responsibilities
Assign Roles and Ownership
Establish Governance
Enable Collaboration & Provide Training and Resources
Monitor and Refine

Figure 6: Steps to Establish Roles & Responsibilities

a) Identify key Stakeholders: To identify the key stakeholders for an SBOM program, organizations should consider representatives from software development, IT operations, security, procurement, business leadership, compliance teams and regulatory bodies.

Include cybersecurity specialists to provide expertise on secure data handling, vulnerability management, and risk assessment.

b) Define SBOM-Related Responsibilities: Outline tasks such as SBOM generation, consumption, vulnerability monitoring, supplier engagement, and secure data management. Assign cybersecurity-focused responsibilities, such as: Classifying SBOM data based on sensitivity and risk, implementing secure SBOM storage and access controls, Integrating SBOM data with vulnerability management and incident response processes

c) Assign Roles and Ownership: Designate a cybersecurity specialist as the SBOM program owner or co-owner to ensure security is embedded throughout. Allocate SBOM responsibilities to stakeholders based on their expertise, with the cybersecurity team playing a key role.

d) Establish Governance: The governance structure should involve key stakeholders from across the organization as discussed in 1st pointer. This governance body would then develop SBOM-specific policies, standards, processes and assign clear accountability, and implement controls to secure the SBOM data.

e) Enable Collaboration: Foster cross-functional collaboration between software, IT, and cybersecurity teams to address SBOM security challenges. Encourage knowledge sharing on secure SBOM practices, emerging threats, and best-in-class security controls.

f) Provide Training and Resources: Offer specialized training on SBOM security requirements, secure data handling, and integrating SBOM with each phase of SSDLC. Equip the team with secure SBOM generation, storage, and consumption tools, as well as vulnerability management and threat intelligence resources.

g) Monitor and Refine: Organizations should conduct regular audits and assessments. This should also involve continuously assessing the SBOM program's security posture and make adjustments to address evolving threats and compliance requirements.

5.2 Roadmap for Navigating the Functions of SBOM

This section explores the goals of the three main aspects of SBOM namely practices, tooling support, and associated issues, aiming to offer Software Developer/Consumer/Integrator organizations a roadmap for navigating the diverse functions and what exactly to be achieved in that specific aspect of SBOM in practice.

Table 7: Objectives for SBOM Concepts

SBOM Functions	Objectives
Benefits	• Improved transparency and visibility into software products should be the primary benefits of SBOM, which forms the foundation for a potential SBOM-centric ecosystem. • The advantages of SBOM should outweigh the costs associated with learning and managing SBOM and their supporting tools.
Adoption	• The third-party (open source or proprietary) components should be equipped with SBOM. • SBOM should be generated for all software products (produced/used) within an organization.
Generation Points	• SBOM should be generated at different stages of the software development lifecycle. • A new SBOM should always be re-generated when there is any change to software artifacts.
Data Fields & standardization	• SBOM should be customized with more organization-specific use cases in terms of data fields and format in addition to an existing minimum number of elements and standard formats.
Distribution	• Generate SBOM for internal use, ensuring proper access control, and consider tailoring content for sharing partial SBOM when distributing proprietary software or components.
Validation	• Supplier should validate SBOM to ensure its integrity.
Vulnerability & Exploitability	• Supplier should provide a Vulnerability Exchange Document to the consumer organization.
Tools	• Integrate SBOM consumption with current tools like vulnerability or configuration management systems.

5.3 Secure SBOM Distribution: Access Control and Public/Private SBOM

To implement access control, precise terms must be defined for SBOM data integration. These terms can be established through licensing, contracts, or other existing mechanisms governing software usage and rights. Suppliers, including open-source maintainers, may prefer public SBOM data, while others might opt for confidentiality, limiting access to select users. By following these steps, organizations should implement a secure and controlled distribution of SBOM, ensuring that sensitive information is accessible only to authorized parties while maintaining transparency and trust in the software supply chain.

5.3.1 Access Control:

5.3.1.1 Define a role-based access control (RBAC) system to manage access to the SBOM data.

5.3.1.2 Identify the different stakeholders (e.g., developers, security teams, supply chain partners) and their respective access requirements.

5.3.1.3 Assign appropriate permissions and privileges to each role, such as Read-only access for general users, Edit and update access for SBOM maintainers, Restricted access for sensitive or confidential SBOM data

5.3.2 Public and Private SBOM:

5.3.2.1 Maintain two versions of the SBOM:

a) Public SBOM: This version contains non-sensitive information that can be shared publicly with all stakeholders.

b) Private SBOM: This version includes sensitive or confidential information, such as vulnerabilities, that should be accessed only by authorized parties.

5.3.3 Secure Distribution Mechanisms:

5.3.3.1 Leverage secure communication protocols, such as HTTPS, to transfer the SBOM data between parties.

5.3.3.2 Implement digital signatures or encryption to ensure the integrity and confidentiality of the SBOM data.

5.3.3.3 Use secure file-sharing platforms or tools that provide access control and audit capabilities.

5.3.4 Automated SBOM Generation and Updates:

5.3.4.1 Integrate the SBOM generation process into the software development lifecycle (SDLC) to ensure the SBOM is up-to-date and accurate.

5.3.4.2 Automate the process of updating the SBOM when changes occur in the software components or dependencies.

5.3.5 SBOM Consumption and Verification:

5.3.5.1 Provide clear guidance and documentation on how to consume and verify the SBOM data.

5.3.5.2 Develop processes and tools to enable stakeholders to validate the SBOM against their specific requirements and security policies.

5.3.6 Monitoring and Auditing:

5.3.6.1 Implement logging and auditing mechanisms to track access and changes to the SBOM data.

5.3.6.2 Regularly review access logs and audit trails to ensure compliance with the defined access control policies.

5.3.7 Incident Response and Remediation:

5.3.7.1 Establish incident response procedures to handle security incidents or breaches related to the SBOM data.

5.3.7.2 Implement processes to quickly assess the impact of vulnerabilities or incidents and coordinate remediation efforts with relevant stakeholders.

5.4 SBOM Sharing

In order to increase the transparency, security and compliance in the software supply chain it is necessary to share the SBOM among the suppliers of the software and the users.

Sharing SBOM documents internally within an organization enables development, security, operations, and legal teams to gain insights into the software components and dependencies used in their projects. This promotes transparency, fosters collaboration, and facilitates compliance with licensing and security requirements. Which in turn will increase the trust among the external partners, suppliers, and vendors. SBOM provides auditable evidence of software composition, licensing, and security measures implemented within a software product or system.

SBOM document sharing can be facilitated through various channels and formats, including:

1. Secure File Sharing Platforms: These platforms should provide a secure and controlled environment for sharing SBOM documents with authorized parties.
2. API Integration: APIs (Application Programming Interfaces) should allow for the automated and secure exchange of SBOM data between different systems or platforms.
3. Collaboration Tools: Collaboration tools, such as project management platforms or document-sharing applications, can facilitate secure SBOM sharing within teams or across organizations.
4. Industry Platforms and Repositories: Several industry-specific platforms and repositories have been established to facilitate the sharing and dissemination of SBOM documents within particular sectors or communities.

   While sharing the documents it is recommended to digitally sign the document for the clients to confirm the authenticity and verify for any tampering. It is also important to identify which SBOM needs to be made public or private while sharing.

6. Vulnerability Tracking and Analysis in SBOM

This chapter discusses vulnerability tracking and analysis using Software Bill of Materials (SBOM) Vulnerability Exploitability eXchange (VEX) and Common Security Advisory Framework (CSAF). VEX facilitates standardized sharing of vulnerability information, while CSAF provides a structured framework for describing security advisories.

a) Design a VEX Document: The Vulnerability Exploitability eXchange (VEX) document should be designed by the organization or entity responsible for managing the software supply chain (e.g. supplier) after a vulnerability is discovered, informing customers about the exploitability status to allow consumers to prioritize their remediation efforts. This should include a team of software developers, vendors, or organizations involved in procurement and compliance responsible for all the tracking and analysis of the vulnerability in the software. It is an iterative process and the VEX document gets updated with each update in the vulnerability including the time taken by the supplier along with remediation, workarounds, restart/downtime required, scores, and risks, the VEX document must include the following about the status of vulnerability in specific software products:

   - Not affected – No remediation is required regarding this vulnerability.
   - Affected – Actions are recommended to remediate or address this vulnerability.
   - Fixed – Represents that these product versions contain a fix for the vulnerability.
   - Under Investigation – It is not yet known whether these product versions are affected by the vulnerability. An update will be provided in a later release.

b) Adoption of Common Security Advisory Framework (CSAF): Subsequently, after the VEX document the supplier should provide the CSAF advisory, which includes detailed information about the vulnerability, such as a description, affected product versions, severity assessment, and recommended mitigation steps. This can be understood by the following example:

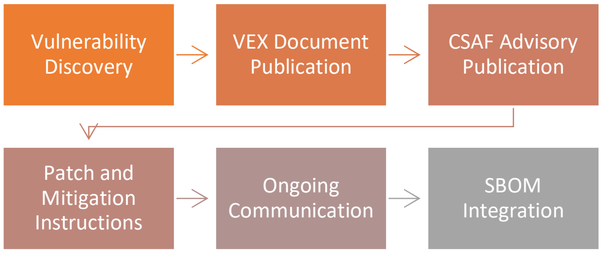


Figure 7: Vulnerability Tracking and Analysis in SBOM Steps Sequence Example

The log4j vulnerability serves as an illustration to map and describe the concept outlined in the figure above.

i. Vulnerability Discovery: In December 2021, a critical vulnerability was discovered in the widely used Log4j logging library.

ii. VEX Publication (1 week): Within a week, the Apache Software Foundation (the maintainers of Log4j) published a VEX document, stating that the vulnerability was "Exploitable".

iii. CSAF Publication (3 weeks): Approximately three weeks after the initial discovery, the Apache Software Foundation released a CSAF advisory with detailed information about the Log4j vulnerability. The CSAF advisory included a description of the vulnerability, affected versions, a CVSS score of 10.0 (critical severity), and mitigation steps.

iv. Patch/Mitigation Instructions: The CSAF advisory provided guidance for users on how to update to a patched version of Log4j or implement other mitigations to address the vulnerability.

v. Ongoing Updates: The Apache Software Foundation continued to monitor the situation and provide updates as new information or additional mitigation strategies became available.

vi. SBOM Integration: Organizations that had the Log4j library included in their software components were able to identify the affected parts of their systems by integrating the VEX and CSAF data into their SBOM. This allowed them to prioritize the remediation efforts and ensure their systems were protected against the Log4Shell vulnerability.

c) Integration with diverse vulnerability databases and advisory: Suppliers and consumers can integrate their SBOM data with vulnerability databases, CERT-In vulnerability notes, alerts, threat intelligence platforms and vendor-specific advisories, enabling comprehensive visibility into their software's security posture. Suppliers directly integrate SBOM data to map components to known vulnerabilities, then provide the enhanced SBOM to customers. Consumers leverage APIs, data feeds, or manual processes to integrate SBOM with vulnerability data, allowing them to identify and prioritize remediation.

d) Implement shift-left approach and vulnerability scanning: Suppliers should implement shift-left vulnerability scanning by integrating security tools into their software development pipeline. This involves automatically analyzing the SBOM data to identify vulnerabilities in the software components during the early stages of the SDLC, such as the build and packaging phases.

7. Recommendations and Best Practices

This chapter delves into practical recommendations and best practices for effectively managing SBOM to enhance software supply chain security.

7.1 Recommendations

7.1.1 All the government, public sector, essential services organizations and organizations involved with software exports and software services industry should include requirements for SBOM in all their software and solutions Purchase/Procurement.

7.1.2 All software supplied to the government; public sector & essential services organizations/departments must be accompanied by a complete SBOM.

7.1.3 All government, public sector and essential services organizations/departments must ensure to maintain SBOM of the software being used, procured and developed.

7.1.4 The SBOM of the software supplied to the government and public sector organizations/departments must include the data fields mentioned in Chapter 4, section 4.2 of this document.

7.1.5 The format to generate the SBOM of the software supplied to government and public sector organizations/departments should be Software Package Data eXchange (SPDX) or CycloneDX.

7.1.6 The software developer/integrator organization that supplies software to government and public sector organizations/departments should design a Vulnerability Exploitability eXchange (VEX) after a vulnerability is discovered informing customers about the exploitability status to allow consumers to prioritize their remediation efforts. The VEX document must include the following about the status of vulnerability in specific software products:

- Not affected – No remediation is required regarding this vulnerability.
- Affected – Actions are recommended to remediate or address this vulnerability.
- Fixed – Represents that these product versions contain a fix for the vulnerability.
- Under Investigation – It is not yet known whether these product versions are affected by the vulnerability. An update will be provided in a later release.

Subsequently, after the VEX document, the supplier should provide the CSAF advisory, which includes detailed information about the vulnerability, such as a description, affected product versions, severity assessment, recommended mitigation steps etc.

7.1.7 Software Developer/Consumer/Integrator organizations should integrate their SBOM data with vulnerability databases, CERT-In vulnerability notes, alerts, threat intelligence platforms and vendor-specific advisories, enabling comprehensive visibility into their software's security posture.

7.1.8 Consumer organizations should update their own SBOM to reflect applied patches or mitigations.

7.1.9 A separate SBOM for each software version, updating it only when additional component information is provided or SBOM errors are corrected.

7.1.10 The consumer organizations (especially the government and public sector organisations) should map and develop an internal SBOM on the basis of the SBOM provided by the supplier.

7.1.11 Security teams of Software consumer organizations should include SBOM inventory in the workflow of vulnerability management.

7.1.12 Regular audits and assessments of SBOM processes should be conducted to ensure accuracy and completeness.

7.1.13 Consumer organizations should combine component data from SBOM with vulnerability status information from VEXes to provide an up-to-date view of the status of vulnerabilities to enable a targeted approach to identifying and addressing software vulnerabilities.

7.1.14 It should be ensured the SBOM data is stored and transmitted securely, using encryption, access controls, and other security measures to protect the sensitive information.

7.1.15 Establish workflows to regularly update the SBOM as new software components are introduced or existing ones are updated.

7.2 Best Practices

7.2.1 Ensure the SBOM captures detailed metadata, such as component names, versions, licenses, and unique identifiers.

7.2.2 Integrate SBOM generation into the secure software development lifecycle (SSDLC) & CI/CD pipelines to maintain the SBOM's accuracy and timeliness

7.2.3 Implement risk-based approaches to prioritize the remediation of vulnerabilities based on factors like severity, exploitability, and potential business impact.

7.2.4 Establish clear policies and procedures for the handling, sharing, and distribution of the SBOM data.

7.2.5 The SBOM data should be generated in such a way that it can be utilized to demonstrate compliance and fulfil regulatory reporting obligations related to software supply chain security.

7.2.6 Implement alerting systems to promptly notify relevant stakeholders about critical security events, enabling timely remediation.

7.2.7 Develop detailed playbooks for responding to security incidents and managing the remediation of vulnerabilities identified through the SBOM analysis.

7.2.8 Adopt a zero-trust security model to verify every user and device trying to connect to the network, enhancing security by eliminating implicit trust assumptions.

7.2.9 Implement Multi Factor Authentication (MFA) mechanisms to add an extra layer of security, reducing the risk of unauthorized access to systems and data.

7.2.10 Conduct periodic vulnerability assessments and measurements to identify and address security weaknesses promptly.

7.2.11 Implement continuous monitoring of software components and dependencies to detect vulnerabilities and address them promptly.

7.2.12 Obtain assurances from third-party software vendors and suppliers regarding the accuracy, completeness, and timeliness of SBOM provided, and establish contractual agreements to ensure compliance with SBOM requirements.

7.2.13 Perform thorough analysis to ensure that the licenses of all software components within an application or software are compatible with each other. Identify any conflicts or restrictions that may arise from combining different licensed components.

7.2.14 Ensure the provision and regular updating of VEX documents alongside CSAF-based advisories with any changes, additions, or updates made to the SBOM.

7.2.15 Provide comprehensive training and awareness programs to educate employees, from developers to security teams, on the importance of SBOM and its role in enhancing software supply chain security.

7.2.16 If the primary component relies on multiple instances with varying meta-information, each instance must be listed separately with its individual meta-information.

8. Quantum BOM (QBOM) & Cryptographic BOM (CBOM)

8.1 What is Crypto & Quantum BOM?

A Quantum Bill of Materials (QBOM) and a Cryptographic Bill of Materials (CBOM) are structured inventories that provide comprehensive visibility into an organization's cryptographic and quantum-related components.

The **QBOM** focuses on components related to quantum computing and quantum-safe cryptography. It includes quantum algorithms, security frameworks, and related technologies, enabling organizations to maintain integrity, compliance, and resilience as they adopt post-quantum cryptographic (PQC) solutions.

The **CBOM** is an inventory of cryptographic assets—including algorithms, keys, protocols, certificates, and dependencies—capturing critical metadata such as usage patterns and expiration dates. It supports effective discovery, management, and reporting of cryptographic components, ensuring transparency and up-to-date security measures.

Together, CBOM and QBOM form the foundation for transitioning to quantum-safe systems. While CBOM facilitates the adoption of quantum-resistant algorithms, QBOM ensures ongoing security and compliance in the quantum era. These inventories empower organizations with the visibility and control required to future-proof their security posture against emerging quantum threats.

8.2 Benefits of Quantum and Crypto BOM

Together, the Quantum Bill of Materials (QBOM) and Cryptographic Bill of Materials (CBOM) form a unified foundation for securing cryptographic systems across classical and quantum domains. By providing complete visibility into both traditional and quantum cryptographic components, QBOM and CBOM jointly enable the following key benefits:

8.2.1 Unified Management of Cryptographic Assets

CBOM and QBOM together offer a comprehensive framework for cataloguing cryptographic and quantum components—such as algorithms, keys, protocols, certificates, hardware, and software libraries—along with critical metadata like versioning, usage, and expiration. This unified inventory enables proactive monitoring, simplified audits, and adherence to evolving security and compliance requirements.

8.2.2 Early Detection of Cryptographic Weaknesses

By analysing data from both BOMs, organizations can identify outdated, misconfigured, or vulnerable cryptographic elements across classical and quantum systems. This joint visibility supports timely remediation, reduces exposure to known threats, and strengthens overall cryptographic hygiene.

8.2.3 Transition to Post-Quantum Cryptography (PQC)

CBOM and QBOM jointly enable organizations to assess their readiness for post-quantum security by identifying dependencies on vulnerable legacy algorithms and mapping paths to adopt quantum-safe alternatives. This supports planning, prioritization, and phased migration toward PQC compliance.

8.2.4 Protection Against Quantum-Era Cyber Threats

The combined inventories help organizations prepare for quantum threats by ensuring that both existing and emerging cryptographic systems are tracked, assessed, and updated in line with quantum-resistant standards. This dual-layer visibility is critical for defending against quantum-enabled attacks.

8.2.5 End-to-End Supply Chain Security

CBOM and QBOM together allow organizations to monitor cryptographic and quantum components across the full supply chain—covering internal deployments and third-party integrations. This reduces the risk of compromised or non-compliant components entering critical systems and ensures enforcement of quantum-safe practices across vendors.

8.2.6 Enhanced Cyber Resilience and Risk Management

With holistic insights, organizations can better identify systemic weaknesses, track cryptographic dependencies, and model potential attack surfaces. This comprehensive perspective strengthens risk management strategies and supports proactive defence against both current and emerging threats.

8.2.7 Strengthened Incident Response and Threat Intelligence

A combined CBOM and QBOM framework equips security teams with the information needed to rapidly identify affected cryptographic or quantum components during cyber incidents. It enhances incident response effectiveness and enables more accurate quantum threat modelling and intelligence gathering.

8.3 Minimum elements of QBOM & CBOM

Table 8: Minimum Elements of QBOM & CBOM

Elements of QBOM	
Element	**Description**
Model Name	Provides a unique identifier for the quantum device or system, ensuring easy reference and distinction.
Version	Tracks version numbers and updates, including new features, security patches, and system enhancements to ensure the latest advancements and fixes are in place.
Vendor & Origin Information	Records the manufacturer and origin of the quantum device, ensuring clarity in vendor communication, support, and maintenance.
License Information	Documents the licensing terms and conditions for the quantum device, ensuring compliance with legal, regulatory, and contractual obligations.
Cryptographic Asset	For a cryptographic asset, the minimum elements that are to be included in the QBOM are mentioned in table 9 depending on the cryptographic asset type.

Elements of QBOM	
Element	**Description**
Communication Protocol	Specifies the communication protocols used by the quantum device, ensuring seamless integration and compatibility with other systems in the infrastructure.
Hardware	Describes the hardware components that make up the quantum device, including the processor, simulators, networking components, and sensors required for quantum operations.
Software Dependencies	Describes the software elements interacting with the quantum hardware, including libraries, APIs, SDKs, and firmware updates.
Environmental Impact	Tracks the environmental impact of the quantum system, including energy consumption and other sustainability factors.
Vulnerabilities	Information about known security vulnerabilities or weaknesses associated with the components, including severity ratings and references to security advisories or CVE identifiers.
Attestations	Digital signature for the QBOM to ensure authenticity and integrity.

Table 9 : Minimum Elements pertaining to Cryptographic Asset

Cryptographic Asset Type	Element	Description
Algorithms	Name	The name of the cryptographic algorithm or asset. For example, "AES-128-GCM" refers to the AES algorithm with a 128-bit key in Galois/Counter Mode (GCM).
	Asset Type	Specifies the type of cryptographic asset. For algorithms, the asset type is "algorithm".
	Primitive	Describes the cryptographic primitive. For "SHA512withRSA", the primitive is "signature" as it's used for digital signing.
	Mode	The operational mode used by the algorithm. For example, "gcm" refers to the Galois/Counter Mode used with AES encryption.

Cryptographic Asset Type	Element	Description
	Crypto Functions	The cryptographic functions supported by the asset. For example, the functions in the case of "AES-128-GCM" are key generation, encryption, decryption, and authentication tag generation.
	Classical security level	The classical security level represents the strength of the cryptographic asset in terms of its resistance to attacks using classical (non-quantum) methods. For AES-128, it's 128 bits.
	OID	The Object Identifier (OID) is a globally unique identifier used to refer to the algorithm. It helps in distinguishing algorithms across different systems. For example, "2.16.840.1.101.3.4.1.6" for AES-128-GCM, "1.2.840.113549.1.1.13" for SHA512withRSA
	List	Lists the cryptographic algorithms employed by the quantum device or system, allowing for an assessment of its security capabilities, especially in the context of post-quantum encryption standards.
Keys	Name	The name of the key, which is a unique identifier for the key used in cryptographic operations.
	Asset Type	Defines the type of cryptographic asset. For keys, the asset type is typically "key".
	id	A unique identifier for the key, such as a key ID or reference number.
	state	The state of the key, such as whether it is active, revoked, or expired.
	size	The size of the key, typically measured in bits. For example, a 128-bit key or a 2048-bit RSA key.
	Creation Date	The date when the key was created.
	Activation Date	The date when the key became operational or was first used.
Protocols	Name	The name of the cryptographic protocol, such as TLS, IPsec, or SSH

Cryptographic Asset Type	Element	Description
	Asset Type	Defines the type of cryptographic asset. In this case, it would be a "protocol"
	Version	The version of the protocol used, such as TLS 1.2 or TLS 1.3.
	Cipher Suites	The set of cryptographic algorithms and parameters supported by the protocol for tasks like encryption, key exchange, and integrity checking.
	OID	The Object Identifier (OID) associated with the protocol, identifying its unique specifications.
Certificates	Name	The name of the certificate, typically referring to its subject or the entity it represents (e.g., a website).
	Asset Type	Defines the type of cryptographic asset. For certificates, the asset type is "certificate".
	Subject Name	This refers to the Distinguished Name (DN) of the entity that the certificate represents. It typically contains information about the organization, domain name
	Issuer Name	The issuer is the Certificate Authority (CA) that issued and signed the certificate. This field contains the DN of the CA that verified and issued the certificate.
	Not Valid Before	This specifies the date and time from which the certificate is valid.
	Not Valid After	This specifies the expiration date and time of the certificate. The certificate becomes invalid after this timestamp.
	Signature Algorithm Reference	This refers to the cryptographic algorithm used to sign the certificate. It provides a reference to the algorithm and its OID (Object Identifier).
	Subject Public Key Reference	This points to the public key used by the subject (the entity being identified in the certificate). It provides a reference to the key's details, including the algorithm.
	Certificate Format	Specifies the format of the certificate. Common formats include X.509, which is the most widely used format for certificates.

Cryptographic Asset Type	Element	Description
	Certificate Extension	This refers to the file extension associated with the certificate. It is commonly .crt for certificates in the X.509 format.

8.4 Recommendations and Best Practices

8.4.1 Recommendations:

8.4.1.1 All government, public sector, and essential services organizations shall require a comprehensive Bill of Materials (BOM), specifically a Cryptographic BOM (CBOM) for cryptographic assets and a Quantum BOM (QBOM) for quantum systems in all related procurements, developments, and integrations.

8.4.1.2 Suppliers of software, systems, or devices involving cryptographic or quantum technologies must provide a complete CBOM and/or QBOM detailing all components, algorithms, technologies, and dependencies present in the delivered solution.

8.4.1.3 Organizations must maintain an accurate and up-to-date CBOM/QBOM for all cryptographic or quantum systems that are used, developed, or procured.

8.4.1.4 The CBOM/QBOM must include detailed transparency into cryptographic and quantum implementations, including hardware, software, algorithms, protocols, and supporting components.

8.4.1.5 BOMs shall be generated using recognized industry-standard formats, such as Software Package Data eXchange (SPDX) or CycloneDX, to ensure interoperability, consistency, and ease of integration.

8.4.1.6 Upon discovery of vulnerabilities in cryptographic or quantum components, developers and integrators must issue a Vulnerability Exploitability eXchange (VEX) document. The VEX must classify vulnerability status as:

a. *Not Affected* – No action required.
b. *Affected* – Recommended actions should be specified.
c. *Fixed* – The vulnerability has been resolved in specific versions.

d. *Under Investigation* – The issue is currently under review.

8.4.1.7 CBOM and QBOM data should be integrated with vulnerability databases, CERT-In advisories, threat intelligence platforms, and vendor bulletins to enable real-time visibility into the security posture of the system.

8.4.1.8 Consumer organizations must create an internal CBOM/QBOM aligned with the supplier's data to ensure accurate representation of deployed components and dependencies.

8.4.1.9 Security teams must incorporate CBOM/QBOM inventories into their vulnerability management workflows to enable continuous monitoring, risk assessment, and remediation planning.

8.4.1.10 Periodic audits and assessments of the CBOM/QBOM must be conducted to verify completeness, accuracy, and compliance with applicable organizational policies and regulatory requirements.

8.4.1.11 Component data from the CBOM/QBOM must be cross-referenced with the VEX status to maintain an up-to-date view of the system's vulnerability landscape and prioritize mitigation actions effectively.

8.4.1.12 CBOM/QBOM data must be stored and transmitted securely using encryption, access control, and data integrity mechanisms to protect sensitive technical and architectural information.

8.4.1.13 Established workflows must ensure that CBOM/QBOMs are updated as new components, cryptographic algorithms, or quantum technologies are introduced, or as existing components are modified or deprecated.

8.4.2 **Best Practices:**

8.4.2.1 Maintain Comprehensive Inventories: Maintain a single, access-controlled inventory for all cryptographic assets and quantum components. The inventory should include accurate descriptions, versioning, configurations, dependencies, and ownership metadata. Access must be limited to authorized personnel to ensure confidentiality, integrity, and accountability.

8.4.2.2 Cryptographic Asset Control: Enforce rigorous cryptographic asset management practices, including the development of complete keymaps from root to leaf. Define secure storage and backup locations for encryption keys. Enforce strict TLS configurations by specifying approved cipher suites, certificate chains, and session parameters, and prohibit the use of weak or deprecated algorithms. Certificate lifecycle management should include automated issuance, rotation, and revocation tracking.

8.4.2.3 Version Control and Change Management: Implement robust version control and change management processes to track all updates and modifications to CBOMs and QBOMs. These workflows should ensure traceability, rollback capability, and integration with secure development pipelines to reflect accurate and real-time system states.

8.4.2.4 Dependency Mapping: Document both the implementation and usage of cryptographic assets. Clearly distinguish between libraries that *implement* cryptographic algorithms and applications that *use* them. This distinction enables a more accurate understanding of component interactions and impact analysis in case of vulnerabilities or updates.

8.4.2.5 Automation: Automate the generation, analysis, and compliance validation of CBOMs and QBOMs using tooling that integrates with development and deployment workflows. Incorporate threat intelligence, vulnerability feeds, and VEX (Vulnerability Exploitability eXchange) inputs to support real-time visibility into the cryptographic and quantum security posture.

8.4.2.6 Risk and Vulnerability Assessment: Conduct periodic risk assessments and cryptographic vulnerability reviews for systems involving cryptographic or quantum components. Leverage adversarial simulations, independent audits, and internal reviews. Findings and mitigations must be documented and linked within the BOM system for traceability.

8.4.2.7 Standards and Compliance Alignment: Ensure that CBOM/QBOM practices align with applicable regulatory and technical standards, including those from NIST (e.g., post-quantum cryptography standards), ETSI, and CERT-

In. Regularly review guidance updates and ensure organizational policies reflect current best practices.

8.4.2.8 Timely Updates: Ensure that any additions, modifications, or deprecations of components, configurations, or cryptographic dependencies are promptly reflected in the CBOM/QBOM. Conduct scheduled reviews, at least quarterly, to verify accuracy and completeness.

8.4.2.9 Cross-Functional Collaboration: Establish collaboration across cybersecurity, engineering, legal, compliance, and vendor management teams to ensure the CBOM/QBOM lifecycle is comprehensively governed. This cross-functional approach ensures security, regulatory compliance, and contractual responsibilities are all effectively addressed.

8.5 Quantum-Readiness and Migration Strategy

8.5.1 Public key systems using RSA, ECC, Diffie-Hellman, and DSA algorithms are vulnerable to Shor's algorithm, which efficiently solves problems that are computationally hard for classical systems but trivial for quantum computers. Organizations must consider transitioning to quantum-resistant cryptographic schemes based on lattice-based, code-based, and multivariate polynomial cryptographic primitives.

8.5.2 To assess quantum readiness, organizations should adopt a risk-based approach involving cryptographic validation testing, independent security assessments, adversarial simulations, and continuous integration of threat intelligence related to quantum advancements.

8.5.3 The implementation of quantum-safe strategies should be guided by a thorough evaluation of threat exposure, technical feasibility, and the organization’s risk tolerance concerning existing cryptographic dependencies that are vulnerable to quantum threats.

8.5.4 Implement tiered vendor Post-Quantum Cryptography (PQC) contractual requirements. Service providers must document all cryptographic implementations (e.g., RSA data encryption, ECC certificate management), provide quarterly migration progress reports with C-level executive attestation, establish penalty clauses for non-compliance, and demonstrate quantum-safe alternatives through proof-of-concepts prior to contract renewals to ensure third-party alignment with quantum readiness goals.

9. Artificial Intelligence Bill of Materials (AIBOM)

9.1 What is Artificial Intelligence Bill of Materials (AIBOM)?

An Artificial Intelligence Bill of Materials (AIBOM) is a comprehensive list of components used in building, training, and deploying AI models. It typically includes hardware (such as servers, sensors, and GPUs), software (AI models, frameworks, and development tools), data sources, and any other essential elements needed for AI implementation.

9.2 Benefits of AIBOM

As AI systems become more intricate, understanding their structure and supply chain is increasingly important. AIBOMs serve several key purposes:

9.2.1 **Security:** AIBOMs help to identify potential vulnerabilities in AI models and their components by providing a detailed breakdown of the system's parts. This enables organizations to pinpoint weak spots, implement appropriate safeguards, and proactively address potential risks before they can be exploited. AI BOM can significantly improve incident response by providing real-time insights into the status of components and materials in the supply chain. It allows for quicker identification of compromised or vulnerable components during a cyberattack.

9.2.2 **Transparency**: By documenting the various components, algorithms, and data sources used in AI systems, AIBOMs provide full visibility into how AI models are built and operate. This transparency fosters greater trust, ensuring that stakeholders understand the decision-making processes behind the AI and the data used to train the models, which is essential in fostering accountability.

9.2.3 **Compliance**: AIBOMs play a vital role in ensuring that AI systems comply with relevant regulatory requirements, industry standards, and ethical guidelines. By mapping out the components and workflows of an AI system, they help organizations demonstrate compliance with laws and regulations, such as data privacy and protection rules, and prevent potential legal or ethical violations.

9.2.4 **Risk Management**: AIBOMs enable organizations to better assess and manage the risks associated with AI systems by providing a clear inventory of all the technological and data-driven elements involved. By identifying dependencies and potential failure points, organizations can put in place the necessary risk mitigation strategies, ensuring that AI deployments are as safe and efficient as possible.

9.3 Minimum Elements of AIBOM

Table 10: Minimum Elements of AIBOM

Element	Description
Model Name	The official name of the AI model, serving as a unique identifier for tracking, referencing, and discussion.
Model Version	The specific version of the AI model, ensuring that any changes, updates, and version control are documented.
Model Type	The type of the model (e.g., text-generation, image-processing, or image-classifier).
Model Developer	The name of the developer or organization responsible for developing the model.
Model Licensing Information	Details of the licenses for the model and any components used (e.g., Apache 2.0, GPL 3.0).
Software Dependencies	All software elements required to run the model, including third-party libraries, frameworks, OS requirements, and proprietary software.
ML Models and Algorithms	The specific machine learning models and algorithms used in the AI system, documenting decision-making processes, input handling, and learning mechanisms.
Model Performance Metrics	Metrics for evaluating the performance of the AI model, such as accuracy, precision, recall, F1 score, etc.
Data Source	The origin or source of the data used to train the model (e.g., proprietary datasets, publicly available datasets, real-time data, synthetic data).

Element	Description
Data Sets	Information about the datasets used to train the model, including names, versions, formats, and limitations. Ensures compliance with privacy and licensing regulations.
Hardware	The hardware or computing resources required to run the model, including processors (e.g., GPUs, TPUs), storage, and memory.
Security Requirements	Encryption methods and access control mechanisms used to protect model data and user information.
Input	The type of input the model accepts, such as text, images, or other data.
Output	The type of output the model generates, such as text, images, or other data.
Intended Usage	The specific use cases or scenarios for which the AI model is designed and intended to be used.
Out of Scope Usage	Uses or scenarios that the AI model is not intended for and should be avoided to prevent misuse or unintended consequences.
Environmental Impact	The potential environmental impact of training and deploying the AI model, including energy consumption, resource use, and carbon footprint.
Vulnerabilities	Known security vulnerabilities or weaknesses in the components, including severity ratings and references to security advisories or CVE identifiers.
Attestations	Digital signature for the model or AIBOM to ensure authenticity and integrity.

9.4 Recommendations and Best Practices

9.4.1 Recommendations:

9.4.1.1 All government, public sector, essential services organizations, and organizations involved in AI-driven development and services should include AIBOM requirements in all AI-related procurements and solutions.

9.4.1.2 All AI solutions supplied to government, public sector, and essential services organizations must be accompanied by AIBOM, detailing the components and dependencies within the AI system.

9.4.1.3 Government, public sector, and essential services organizations must ensure that an AIBOM is maintained for all AI systems being used, procured, and developed.

9.4.1.4 The AIBOM of AI systems supplied to government and public sector organizations must include the elements specified in this document ensuring transparency regarding the AI model's components and dependencies.

9.4.1.5 The format for generating AIBOM for AI solutions should adhere to established standards, such as Software Package Data eXchange (SPDX) or CycloneDX, ensuring compatibility and uniformity across the industry.

9.4.1.6 The AI developer/integrator organization that supplies AI solutions to government and public sector organizations should develop a Vulnerability Exploitability eXchange (VEX) for AI models after a vulnerability or security issue is discovered. This will inform stakeholders about the exploitability status, enabling prioritized response. VEX document categories should include:

- Not Affected – No remediation is required.
- Affected – Recommended actions to address the vulnerability.
- Fixed – A fix is available in product versions.
- Under Investigation – Ongoing investigation into whether the vulnerability affects the product.

9.4.1.7 AI developer/consumer/integrator organizations should integrate their AIBOM data with vulnerability databases, CERT-In vulnerability notes, alerts, threat intelligence platforms, and vendor-specific advisories. This integration ensures real-time visibility into AI system security.

9.4.1.8 Consumer organizations (especially government and public sector bodies) should develop an internal AIBOM based on the one provided by the AI solution supplier, ensuring full alignment with the system's components and their dependencies.

9.4.1.9 Security teams in consumer organizations should incorporate AIBOM inventory into their vulnerability management workflows to ensure continuous monitoring and proactive mitigation of AI system vulnerabilities.

9.4.1.10 Regular audits and assessments of the AIBOM process should be conducted to ensure its accuracy, completeness, and compliance with organizational and regulatory standards.

9.4.1.11 Consumer organizations should combine component data from AIBOM with vulnerability status information from VEXes to provide an up-to-date view of the AI system's vulnerability landscape, enabling targeted remediation.

9.4.1.12 AIBOM data should be securely stored and transmitted using encryption, access control mechanisms, and other security measures to protect the sensitive AI model information.

9.4.1.13 Establish workflows to regularly update the AIBOM as new AI components or models are introduced, or existing ones are updated, ensuring that the AIBOM remains accurate throughout the lifecycle of the AI system.

9.4.2 Best Practices:

9.4.2.1 Maintain a Comprehensive Inventory: The first step in creating AIBOM is to compile a comprehensive inventory of all the critical components that make up AI system. It ensures that all components and their dependencies are thoroughly accounted for, offering a clear and organized overview of AI system.

9.4.2.2 **Standardized Format**: Use standardized formats, like CycloneDX or SPDX, to ensure consistency and interoperability across systems.

9.4.2.3 **Automate the AIBOM Generation:** To improve efficiency, consider automating the AIBOM creation process as part of model development and deployment pipelines. Automation ensures consistency and helps keep AIBOM up to date with every iteration of your models.

9.4.2.4 **Foster Reproducibility:** To ensure others can replicate results and verify AI models, include detailed scripts, model weights, and configuration settings in AIBOM. This enhances reproducibility, helping others verify the accuracy of models and encouraging further collaboration.

9.4.2.5 **Prioritize Critical Models:** Focus your initial AIBOM efforts on the high-priority or high-risk models within your AI system. By targeting these areas first, you'll have the greatest impact on ensuring that system is secure, compliant, and well-optimized.

9.4.2.6 **Track Model Lineage:** As models evolve, it's essential to maintain clear records of their development. This includes tracking:

- Model Versions
- Retraining Activities
- Modifications and Enhancements

By maintaining model lineage, a clear history of how the AI system has changed over time is provided, ensuring accountability and facilitating reproducibility.

10. Hardware Bill of Material (HBOM)

10.1 What is HBOM?

Hardware Bill of Materials (HBOM) refers to a structured inventory of all physical components, sub-components, embedded devices, and associated materials that constitute a hardware system or product. This HBOM provides a structured inventory of hardware components including servers, networking equipment, storage devices, end-user systems, power backup solutions, and security infrastructure. In the context of cybersecurity and supply chain risk management, the HBOM is a critical element for ensuring traceability, integrity, and compliance with applicable security and procurement standards.

10.2 Benefits of HBOM

10.2.1 Supply Chain Risk Management: An HBOM offers comprehensive visibility into the origin, authenticity, and integrity of each hardware component used in a system. This traceability helps organizations manage supply chain risks by identifying trusted suppliers and detecting counterfeit or unapproved parts. It also enables better vendor evaluation and risk-based decision-making.

10.2.2 Improved Cybersecurity Posture: Maintaining an HBOM supports the identification of hardware components that may be vulnerable, insecure, or compromised. It allows security teams to detect malicious implants or unauthorized modifications and strengthens the overall security posture by enabling device-level integrity verification.

10.2.3 Effective Asset Lifecycle Management: With a complete HBOM, organizations can manage hardware assets more efficiently throughout their lifecycle—from procurement and deployment to decommissioning. It helps in tracking component versions, planning for upgrades, and addressing end-of-life or unsupported hardware well in advance.

10.2.4 Integration with Software Bill of Materials (SBOM): When used in conjunction with an SBOM, the HBOM provides holistic visibility into both hardware and software components of a system. This full-stack transparency is critical for secure system design, firmware assurance, and effective vulnerability management across all layers of technology.

10.2.5 **Business Continuity and Operational Efficiency:** An HBOM supports faster procurement and replacement of hardware components during failures or disruptions. By knowing the exact specifications and sources of components, organizations can reduce downtime, improve spare management, and optimize maintenance operations.

10.2.6 **Supports Secure-by-Design Principles:** The HBOM aligns with secure-by-design practices by ensuring that all hardware components are verified, approved, and fit for deployment in trusted environments. It contributes to establishing hardware trust anchors, which are essential in building secure architectures and systems from the ground up.

10.3 Minimum elements of HBOM

Table 11: Minimum Elements of HBOM

Element	Description
Product Name	The common or marketing name of the hardware item.
Product Version	The specific iteration or release number of the product.
Product Details	Key specifications and features that describe the hardware.
Warranty/AMC	Information regarding the product's guarantee and any annual maintenance contract.
Manufacturer Name	The name of the company responsible for producing the hardware.
Manufacturer Location	The geographical address of the product's manufacturer.
Manufacturing Date	The specific date when the product was produced.

Element	Description
Supplier Information	Details about the company that provided or sold the product
Supplier Location	The geographical address of the product's supplier.
Model Number	A specific alphanumeric identifier for a particular product design.
Serial Number	A unique identifier assigned to each individual unit of a product for tracking.
Technical Specification	Detailed technical characteristics and performance parameters of the component (e.g., voltage, frequency, capacity).
Supplier Information	The name and contact details of the company that supplied the component to the manufacturer of the larger product.
Supplier Location	The geographic location (city, country) of the direct supplier of the component.
Technology Node	For integrated circuits, this refers to the semiconductor manufacturing process technology (e.g., 7nm, 14nm), indicating its feature size and generation.
Compliance	Declarations or certifications indicating adherence to relevant industry standards, regulations, or environmental directives (e.g., RoHS, CE).
Power supply	Details about the component's power requirements, such as voltage, current, and wattage.
License Information	Any associated intellectual property licenses or usage terms, particularly relevant for firmware or embedded software within the component.
Test Result	Information about the component's performance during quality assurance or functional testing, indicating its operational status.
Sub-component	This refers to the recursive nature of an HBOM; if a component itself contains further distinct hardware parts, those would also be listed with similar detailed attributes.

10.4 Recommendations & Best Practices

10.4.1 Recommendations

10.4.1.1 All government, public sector, essential services organizations, and organizations involved in manufacturing, deploying, or integrating hardware products should mandate HBOM requirements in all hardware procurement and supply contracts.

10.4.1.2 All hardware supplied to government, public sector, and essential services organizations must be accompanied by a complete and accurate HBOM that details all critical hardware components and subcomponents.

10.4.1.3 Government, public sector, and essential services organizations must maintain HBOMs for all hardware systems in use, including those procured externally or developed in-house, to support effective asset management and risk assessment.

10.4.1.4 The HBOM accompanying hardware supplied to government and public sector entities must include data fields such as manufacturer name, model number, component version, firmware version, origin, criticality rating, and associated vulnerabilities.

10.4.1.5 Hardware vendors and integrators must provide a **Vulnerability Exploitability Exchange (VEX)** or equivalent hardware vulnerability statement when a component vulnerability is discovered. This should include:

•**Not affected** – The hardware is not impacted by the identified vulnerability.

•**Affected** – The hardware is vulnerable and needs mitigation.

•**Fixed** – The vulnerability has been addressed in updated hardware/firmware versions.

•**Under Investigation** – Evaluation is ongoing; updates to follow.

Subsequently, a CSAF advisory should be provided with detailed technical information, severity, mitigations, and impact assessment.

10.4.1.6 The recommended formats to generate HBOMs should align with structured and machine-readable standards, preferably using extended SBOM formats like CycloneDX or custom XML/JSON schemas based on organizational needs.

10.4.1.7 Organizations involved in hardware design, supply, or consumption should integrate HBOM data with hardware-focused vulnerability databases, CERT-In vulnerability notes, firmware advisories, and threat intelligence platforms to enable holistic supply chain visibility.

10.4.1.8 Consumer organizations must update their internal HBOMs to reflect any patching, component replacement, or firmware updates applied to the hardware.

10.4.1.9 A separate HBOM must be maintained for each hardware version or model. Updates should be made only when new component information is available or corrections are necessary.

10.4.1.10 Consumer organizations (especially in the government and public sector) should create an internal HBOM based on the one provided by the supplier, enriched with asset-specific details.

10.4.1.11 Security teams of hardware-consuming organizations should incorporate HBOM inventory into their asset management and vulnerability management workflows.

10.4.1.12 Periodic audits and assessments should be conducted to ensure HBOM accuracy, completeness, and traceability across the hardware lifecycle.

10.4.1.13 Organizations should correlate HBOM data with vulnerability intelligence (from VEX/CSAF or firmware CVEs) to continuously assess and prioritize hardware-related risk exposure.

10.4.1.14 HBOM data must be securely stored and transmitted using encryption, access controls, and data integrity mechanisms to prevent tampering or unauthorized access.

10.4.1.15 Workflows must be established to routinely update HBOMs as new components are added, replaced, or modified during the lifecycle of the hardware system.

10.4.2 Best Practices

10.4.2.1 Include Metadata for each Component: Effective supply chain management is crucial for maintaining security and compliance. It enables precise vulnerability tracking, which is vital for promptly addressing issues like compromised firmware versions. Furthermore, robust supply chain practices support compliance with national import/export or security regulations, ensuring legal and secure international trade. Finally, such vigilance allows security teams to thoroughly assess risks tied to manufacturing origins, especially concerning geopolitically sensitive areas, thereby mitigating potential threats to the integrity and reliability of components.

10.4.2.2 Maintain Version Control & Change Logs: Establishing thorough component tracking ensures traceability in the event of an incident, supports forensic analysis should a malicious component be discovered, and allows for regulatory audits and validation of declared hardware.

10.4.2.3 Ensure Data Confidentiality and Access Control: HBOMs contain highly sensitive information regarding product design, making their protection crucial for preventing supply chain attacks that could arise from insider threats or third-party compromises. By diligently managing and securing HBOMs, organizations can significantly reduce their vulnerability to such attacks, thereby ensuring compliance with intellectual property protection and various cyber laws.

10.4.2.4 Conduct Regular Audits and Validation: Implementing advanced component analysis is crucial for several reasons: it detects hidden or undeclared components, thereby uncovering potential supply chain backdoors; it validates trust in third-party suppliers by verifying the authenticity and integrity of their provided parts; and ultimately, it enables regulatory or contractual enforcement by providing verifiable proof of compliance or deviation.

10.4.2.5 Automate HBOM Generation: Automating the process of generating and maintaining HBOMs offers significant advantages. It reduces human error, ensuring accuracy and consistency. Furthermore, it keeps HBOMs up-to-date with actual production builds, eliminating discrepancies between documentation and physical products. Ultimately, this automation saves considerable time and effort in documentation and audit readiness, streamlining operations and improving compliance.

10.4.2.6 Track Provenance and Supply Chain Risk: Implementing robust component verification processes offers several key benefits. Primarily, it prevents the use of counterfeit or unauthorized components, safeguarding product quality and integrity. Additionally, such processes help identify potential vendor lock-in or critical dependencies within the supply chain, allowing for proactive mitigation strategies. Finally, they support comprehensive supplier risk scoring, encompassing factors like geopolitical stability, financial health, and cybersecurity posture, leading to a more resilient and secure supply chain.