

(Provisional Translation)

Guidance on Introduction of Software Bill of Materials (SBOM) for Software Management

ver. 2.0

Cybersecurity Division
Commerce and Information Policy Bureau
Ministry of Economy, Trade and Industry

August 29, 2024

Table of Contents

1. Background and objectives.....	3
1.1. Background	3
1.2. Objectives	6
1.3. Main target readers	7
1.4. Main target software.....	8
1.5. How to use.....	8
1.6. Summary of this Guidance.....	10
2. Overview of SBOM	13
2.1. What is SBOM?	13
2.2. Benefits of SBOM.....	16
2.3. “Minimum Elements” of SBOM.....	23
2.4. SBOM formats (Examples)	25
2.5. Myths and facts.....	34
3. Basic guidance and overall view on SBOM introduction.....	38
3.1. Basic guidance for SBOM introduction.....	38
3.2. SBOM introduction process	38
4. Environment and system development phase.....	40
4.1. Clarification the scope of the SBOM application	40
4.2. SBOM tools selection.....	45
4.3. SBOM tools installation	52
4.4. Learning about SBOM tools	54
5. SBOM production and sharing phase.....	55
5.1. Component analysis.....	55
5.2. SBOM production.....	60
5.3. SBOM sharing	62
6. SBOM use and management phase	64
6.1. Vulnerability management, license management, etc.....	64
6.2. SBOM information management	67
7. Specification of Vulnerability Management Process	69
7.1. Purpose	69

7.2.	Challenges and issues in vulnerability management.....	69
7.3.	Overview of the entire process	70
7.4.	Procedures and methods for each phase	72
8.	Appendix: SBOM Compliance Model	103
8.1.	Purpose and background	103
8.2.	SBOM visualization framework and Compliance Model.....	105
8.3.	SBOM Compliance Model and utilization methods	114
8.4.	Reference example of SBOM Compliance Model (Automotive Sector)...	119
8.5.	Reference example of SBOM Compliance Model (Software Product Sector)	130
8.6.	Reference example of SBOM Compliance Model (Medical Device Sector)	139
8.7.	Cross-sector comparison of the SBOM Compliance Models (Draft).....	152
9.	Appendix: SBOM Contract Model.....	154
9.1.	Background and purpose (problem awareness).....	154
9.2.	Overview	155
9.3.	Concept of the Contract Model.....	156
9.4.	SBOM Contract Model.....	157
9.5.	Relationship and positioning of the SBOM Compliance Model and the SBOM Contract Model	160
9.6.	Relationship with existing model contracts	161
9.7.	Utilization patterns	162
9.8.	Challenges and future directions for consideration	163
10.	Appendix	164
10.1.	Checklist of actions for the introduction of SBOM.....	164
10.2.	Glossary.....	168
10.3.	Reference information	172

1. Background and objectives

1.1. Background

As industrial activities become more service-oriented, the importance of software in industry is increasing. In recent years, software has been increasingly implemented to control industrial machinery, automobiles, etc. In IoT devices and services and 5G technology, a variety of added values are expected to be created by building hardware systems with general-purpose devices and then adding various functions through software.

To ensure the safety and security of software used by consumers, as well as software used by companies, it is necessary to properly manage the vulnerabilities of that software. Even if the software is configured not to contain vulnerabilities in the planning and design stages, vulnerabilities may be discovered after the product is shipped. In such cases, the party utilizing the software is required to update the software and take other measures. In addition, when maintenance and support end for software used in the company's products and services, the company is required to consider the management of vulnerabilities discovered thereafter, including the possibility of changing to alternative software. However, as the software supply chain becomes more complex and the use of open-source software (OSS) becomes more common, it is difficult to know what kind of software is included as a component, even if the software is used in the company's own products. Many organizations manage the software used in their IT systems as assets, but only the upper-level components directly used by developers are subject to asset management, while many of the lower-level components that are indirectly used within the directly used components are not subject to asset management. Therefore, when vulnerabilities are discovered in components such as OSS that are used as lower-level components, it is not possible to determine the effects of indirect vulnerabilities by simply comparing vulnerability information with the asset management ledger.

Software Bill of Materials (SBOM) has been attracting attention as a method to solve the problems of both software developers and users regarding software vulnerability management. An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships. The SBOM may contain the name and version information of the components included in the software, the developer of the components, and other information, and contain information about proprietary

software as well as OSS. The mutual sharing of SBOM across organizations from upstream to downstream in the software supply chain is expected to increase the transparency of the software supply chain, and to be one solution to the issue of component vulnerability management. SBOM began attracting attention through Proof-of-Concepts (PoC) launched in July 2018 by the U.S. Department of Commerce's National Telecommunications and Information Administration (NTIA) and has been increasingly popular worldwide, with an executive order signed by U.S. President Biden¹ in May 2021. A survey conducted by the Linux Foundation of 412 global organizations in the third quarter of 2021² found that 48% of organizations surveyed have deployed an SBOM. The Linux Foundation estimates that the adoption rate will be 78% in 2022 and 88% in 2023, based on the SBOM readiness and planning status of the surveyed organizations.

In Japan, the Ministry of Economy, Trade and Industry (METI) established the Task Force for Evaluating Software Management Methods, etc. toward Ensuring Cyber/Physical Security (Software Task Force) in September 2019, and has since made extensive discussions on software management methods including SBOM. Through the discussions of the Software Task Force, the following issues were raised as essential considerations for the implementation of SBOM: cost-effectiveness of SBOM introduction, issues related to sharing SBOM in the supply chain, issues related to contracts when managing SBOM, and issues related to the implementation of SBOM in small and medium-sized enterprises. Considering these issues, METI conducted PoC from 2021 for SBOM introduction and beyond and evaluated the costs and benefits of SBOM introduction in several industrial sectors. The FY2021 PoC targeted software for automated driving system development, while the FY2022 PoC focused on dental CTs in the medical device field, heater controllers in the automotive field, and network threat detection software in the software field. Through these PoC, the following benefits and effects of SBOM were confirmed, especially the benefits for software vulnerability management and license management, which may result in the benefit of increased development productivity.

- Comparing the workloads for manual component management and the

¹ Executive Order on Improving the Nation's Cybersecurity

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

² Linux Foundation, The State of Software Bill of Materials (SBOM) and Cybersecurity Readiness

<https://www.linuxfoundation.org/tools/the-state-of-software-bill-of-materials-sbom-and-cybersecurity-readiness/>

workloads for management using an SBOM, the workloads for the latter are smaller. When implementing an SBOM, the initial workloads are large, but the burden can be reduced by using SBOM tools³.

- By creating and managing an SBOM, it is possible to shorten the lead time needed to identify the impact of vulnerabilities in software components when they are discovered, eventually leading to a reduction in the risk of residual vulnerabilities and in workloads required to respond to vulnerabilities. By utilizing commercial SBOM tools, dependencies between different pieces of OSS and recursive use of OSS (reused components) can also be efficiently detected and managed.
- By creating and managing an SBOM, 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. In particular, the SBOM tool enables more efficient license management because it allows users to utilize functions for compliance, such as displaying the contents of each license and warning of licenses that require attention.

On the other hand, the following issues were identified regarding SBOM introduction:

- If the overall configuration of the target software system is not understood, the scope of application of the SBOM tool cannot be properly set and effective risk management cannot be implemented.
- Workloads are required to learn and develop the environment to implement SBOM tools.
- OSS SBOM tools require many workloads to implement due to a lack of information about environmental maintenance and learning. In addition, there are many things to be aware of when using such tools, such as insufficient detection of recursively used components, limitations on SBOM formats that can be handled, and license false negatives.
- Simply applying an SBOM tool may result in undetected components in the target software.
- The output results of the SBOM tool need to be scrutinized, as there may be

³ In this Guidance, tools that can create, share, utilize, and manage an SBOM are collectively referred to as "SBOM tools," which are sometimes called SBOM management tools, OSS management tools, software configuration analysis (SCA) tools, etc.

cases of component false positives, false negatives, and erroneous vulnerability information.

- The workloads required to scrutinize the output of SBOM tools are significant because the internal configuration and technologies used for third-party components are not known.
- Currently, there are few SBOM tools that can read SBOM generated by different SBOM tools and use them for vulnerability management, making it difficult to mutually share an SBOM among different SBOM tools.
- It is difficult to determine which vulnerabilities need to be addressed for a given component and which ones shall be addressed first.
- There is a lack of information on specific methods for managing vulnerabilities using SBOM, including issues such as vulnerability identification and response prioritization.
- There is a lack of information on contractual matters to clarify SBOM-related requirements and responsibilities between the purchaser and supplier in software development outsourcing and procurement of off-the-shelf products.

In summary, while it was confirmed that SBOM can be used for efficient software management, it was also clear that there are various issues that need to be addressed when implementing an SBOM.

1.2. Objectives

To solve various issues related to the creation, sharing, operation, and management of SBOM for software management, this Guidance provides basic information about SBOM, including an overview of SBOM and the benefits of SBOM introduction, and presents a series of processes for SBOM introduction, including the establishment of an environment and system for SBOM creation, SBOM creation and sharing, and SBOM operation and management, at software suppliers. It also shows the main implementation items in each phase and the key points that companies shall be aware of when implementing an SBOM to support efficient and effective SBOM introduction. In addition, it is shown how to visualize the scope of SBOM coverage and the requirements for contracts. Although this Guidance is intended primarily for software suppliers, it can also be used and referenced by companies that procure and use software. As SBOM is a method of software

management, the aim should not be to create an SBOM itself, but to use SBOM to achieve appropriate software management. In response to the recent trend of increased use of OSS in software development, OSS management is also important in establishing software security measures. The Ministry of Economy, Trade and Industry (METI) has published “Collection of Use Case Examples Regarding Management Methods for Utilizing Open-Source Software and Ensuring Its Security”⁴ as part of documentation concerning OSS management. It is recommended to refer to this document as well as this Guidance.

1.3. Main target readers

This Guidance mainly targets departments involved in software security at software suppliers, such as development/design departments and departments in charge of product security (PSIRT, etc.), as well as in management. For departments involved in software security, this Guidance describes the process of SBOM introduction, the main items for SBOM introduction, and points to note when implementing an SBOM for software management. If it is believed that management is not fully aware of SBOM as a method of software management, it is expected to use “1.6 Summary of this Guidance” to communicate appropriately with management. For management, this Guidance presents the effects and benefits of SBOM and the misconceptions and facts about SBOM that can be referred to when making decisions regarding SBOM introduction. When making decisions regarding the SBOM introduction, it is expected that the contents of “1.6 Summary of this Guidance” will be well understood.

“Section 8 (Appendix) SBOM Compliance Model” is intended for development and operations departments and security departments (PSIRT) as suppliers of software and SBOM, and for users, procurement departments of development companies, development departments, quality assurance departments, and security departments as purchasers of software.

The main intended readers of “Section 9 (Appendix) SBOM Contract Model” are legal staff and developers involved in transaction contracts that stipulate the requirements, responsibilities, cost burdens, etc. related to SBOM for those placing

⁴ Ministry of Economy, Trade and Industry: Collection of Use Case Examples Regarding Management Methods for Utilizing Open Source Software and Ensuring Its Security
https://www.meti.go.jp/policy/netsecurity/wg1/ossjirei_20220801.pdf

and receiving orders for software.

This Guidance is mainly intended for those who are new to SBOM, such as organizations that are not yet aware of the details of their efforts to implement SBOM. Related to the above, the content of this Guidance concerning license management can be used by the legal and intellectual property departments of an organization. Furthermore, its general content can be used in part by companies that procure and use software, not just software suppliers. SBOM are used for a variety of purposes, including vulnerability management, license management, export management, and patent management, but this document focuses on vulnerability management and license management.

1.4. Main target software

This Guidance describes the process for implementing an SBOM, mainly for packaged software and embedded software, as well as the main implementation items in each process and points to be aware of when introducing an SBOM. While it is important to manage components that include hardware vulnerabilities, this guidance focuses on software.

1.5. How to use

Organizations implementing an SBOM are expected to recognize the basic information about SBOM and confirm the process for SBOM introduction based on this Guidance. It is also expected that organizations will proceed with the implementation of an SBOM while confirming the main implementation items in each step and the points that shall be recognized when implementing the SBOM.

Section 2 provides an overview of the basic SBOM-related issues. Section 3 provides an overview of the basic guidelines for introducing SBOM and the overall introduction process. Section 4 provides an overview of the implementation issues related to the initial introduction of SBOM. Sections 5 and 6 provide an overview of the implementation issues related to the SBOM creation and sharing phase and the operation and management phase for each project after the initial introduction. Since there are issues with the current vulnerability DB environment when using SBOM to identify and prioritize vulnerabilities, it is expected that organizations will select and customize a method that suits their organization, referring to the

solutions and know-how presented in Section 7. Section 8 (Appendix) can be used as a framework for visualizing the scope of SBOM and indicating the management level. Section 9 (Appendix) can be used as a reference for clearly stipulating the requirements and responsibilities of the parties involved in a contract. Section 10.1 (Appendix) provides a checklist of the items to be implemented at each step of the SBOM introduction process, and it is recommended that this be referred to when acting toward SBOM introduction.

1.6. Summary of this Guidance

《Key points of this Guidance》

- Software security threats that can affect business operations have increased dramatically in recent years.
- The Software Bill of Materials (SBOM), a method of software management, is attracting attention in response to the threats, and the number of companies adopting it is increasing worldwide.
- SBOM can reduce the risk and cost of managing software vulnerabilities and licenses.
- It is expected that this Guidance will be used to accelerate efforts to implement SBOM for software management.

《Background and outline of this Guidance》

[Threats to the software supply chain]

- ✓ As software supply chains become more complex and the use of open-source software (OSS) becomes more common, security threats to software have increased dramatically in recent years. Apache Log4j vulnerabilities discovered in December 2021 have had a worldwide impact. According to data, from 2019 to 2022, the average annual growth rate of software supply chain attacks reached 742%.
- ✓ Software security threats have a significant impact on business operations. For example, average companies affected by SolarWinds cyberattacks lost approximately 11% of their annual revenue, and in some cases, remaining vulnerabilities in products have led to product recalls and sales suspensions.
- ✓ In response to increasing threats to software, it is important to implement software management efficiently and effectively, such as properly managing vulnerabilities contained in software and promptly responding to vulnerabilities when they are revealed.

[Benefits of using SBOM in software management]

- ✓ The Software Bill of Materials (SBOM) has been attracting attention as a method for efficiently managing software developed through the supply chain. The SBOM is a machine-processable list that includes information about software components and their dependencies. The number of companies implementing an SBOM is increasing worldwide. Regulations and institutionalization are also beginning to be considered, with SBOM being recommended in some fields,

such as the medical device sector.

- ✓ While software management requires an enormous amount of information, the implementation of a machine-readable SBOM can reduce the cost and workload required for software management, which in turn leads to higher development productivity. In fact, in a Proof-of-Concept (PoC) conducted in the medical device sector by the METI, vulnerability management using SBOM reduced management workloads by about 70% compared to manual management.
- ✓ In addition, as a benefit to vulnerability management, the creation and ongoing management of SBOM is expected to increase software transparency and reduce the risk of residual vulnerabilities, as well as increase the efficiency of vulnerability response through the supply chain.
- ✓ For this reason, it is hoped that introducing SBOM through the supply chain and effectively sharing the burden of components management and vulnerability response between the companies involved in development and operation will eliminate the burden on vendors and improve efficiency overall.
- ✓ Furthermore, as an advantage in license management, SBOM will help reduce the risk of license violations by managing OSS license information.

[Points to using this Guidance]

- ✓ To support efficient and effective SBOM introduction by companies, this Guidance provides basic information about SBOM and presents the main implementation items for SBOM introduction and points to be aware of when implementing an SBOM.
- ✓ For efficient and effective software management, it is expected that management will use this Guidance to make decisions regarding the implementation of the SBOM and that departments involved in software security will take concrete steps for SBOM introduction.

Column: Key indices for software security threats

Security threats to software have grown in recent years as software supply chains become more complex and the use of OSS becomes more common. Below are some key figures that illustrate the current state of software security threats in recent years.

84%: Percentage of code bases containing vulnerabilities

According to a survey of 1,700 codebases published by Synopsys in 2023, the percentage of codebases containing OSS was 96%. Of those, 84% of the codebases contained at least one vulnerability⁵.

62% : Percentage of companies that suffered software supply chain attacks in 2021

According to a survey published by Anchore in 2022 that covered 428 companies in North America, the EU, and the UK, 62% of companies were affected by software supply chain attacks in the past year⁶.

↑ 742% : Average annual increase in software supply chain attacks from 2019 to 2022

According to a study published by Sonatype in 2023, the average annual increase in software supply chain attacks over the three-year period from 2019 to 2022 was 742%, exceeding 88,000 attacks in 2022. With 216 attacks from February 2015 to June 2019, the number of software supply chain attacks has increased exponentially in recent years⁷.

↓ 11% : Impact of the SolarWinds cyberattack on company revenues

According to IronNet's 2021 survey of 473 companies in the U.S., U.K., and Singapore, 85% of the companies were affected by SolarWinds cyberattacks, which cost them on average about 11% of their annual revenue⁸.

⁵ Synopsys, 2023 Open Source Security and Risk Analysis Report

<https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html>

⁶ Anchore, 2022 Security Trends: Software Supply Chain Survey

<https://anchore.com/blog/2022-security-trends-software-supply-chain-survey/>

⁷ Sonatype, 8th Annual State of the Software Supply Chain Report

<https://www.sonatype.com/state-of-the-software-supply-chain/implementation>

⁸ IronNet, 2021 Cybersecurity Impact Report

<https://www.ironnet.com/hubfs/IronNet-2021-Cybersecurity-Impact-Report-June2021.pdf>

2. Overview of SBOM

2.1. What is SBOM?

The SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships. The SBOM may contain the name and version information of the components included in the software, the developer of the components, and other information, and contain information about proprietary software as well as OSS. The mutual sharing of SBOM across organizations from upstream to downstream in the software supply chain is expected to increase the transparency of the software supply chain, and to be one solution to the issue of component vulnerability management.

To flesh out the SBOM, consider the following simplified scenario:

- Company A developed a software named Application using two components—Company B's Browser and Community P's Protocol.
- Company B's Browser uses a component of Compression Engine developed by Mr. C.
- Company B created its own SBOM for the browser and shared it with Company A. However, because Company A was unable to obtain the SBOM information about Mr. C's and Community P's components, Company A created an SBOM of Mr. C's and Community P's components.

The relationships among the players and components in this scenario can be represented as shown in Figure 2-1. As shown in this figure, many SBOM entities play the role of software suppliers as well as consumers of the SBOM shared with others. That is, in addition to utilizing information in an SBOM obtained from another entity, the first entity may also play a role in creating an SBOM related to the newly developed components and sharing the SBOM with other entities. Ideally, the supplier of a software component shall also be the author of the corresponding SBOM, but this is not always the case in the current situation where SBOM are not yet completely widespread. In this scenario, since Company B created the SBOM in-house, the supplier of a browser component and the SBOM author for the component are the same. In the case of a protocol, however, since Community P did not create the SBOM, but Company A did, then the supplier is Community P,

while the SBOM author is Company A.

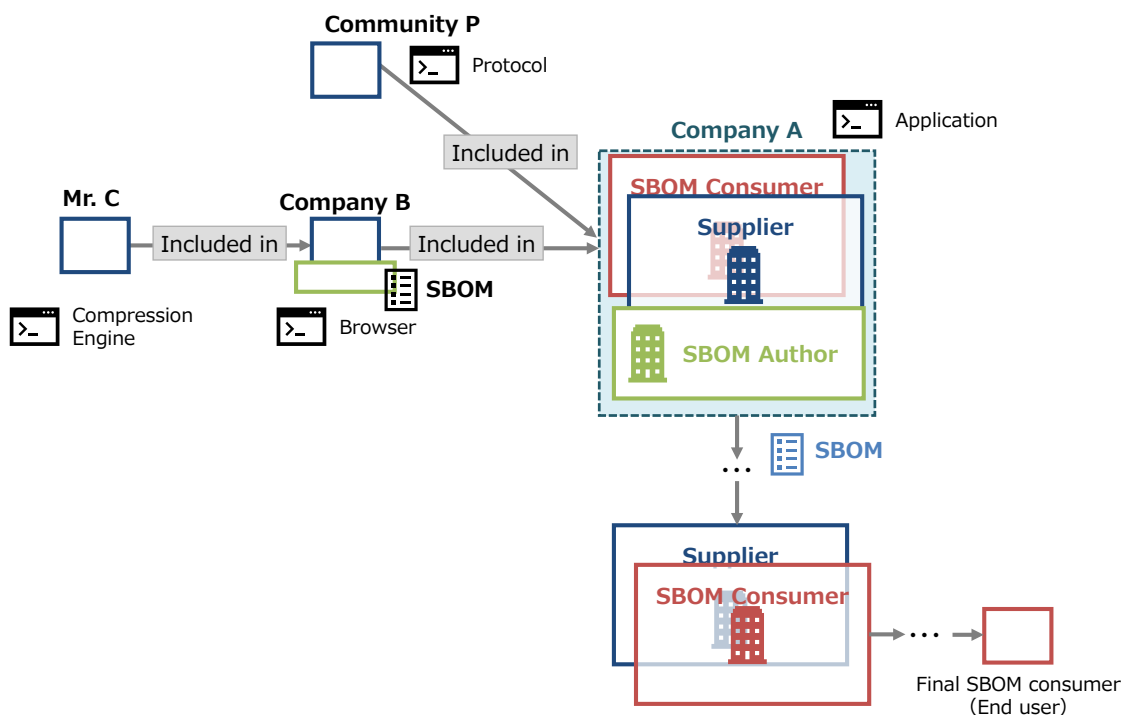


Figure 2-1 Relationship between players in the scenario

In the above scenario, the conceptual image of the SBOM to be created by Company A is given in Table 2-1. This image lists its supplier, version, component name, SBOM author, etc. for each component. By creating an SBOM, it is possible to identify and manage when and by whom each component was developed, what implementation it has with other components, and who created the SBOM for that component. When a vulnerability in a particular component is revealed, this allows the system to immediately recognize which components are affected by the vulnerability, allowing for a quick response to the vulnerability. The mutual sharing of an SBOM across organizations will make information about each component visible and contribute to improving the transparency of the software supply chain.

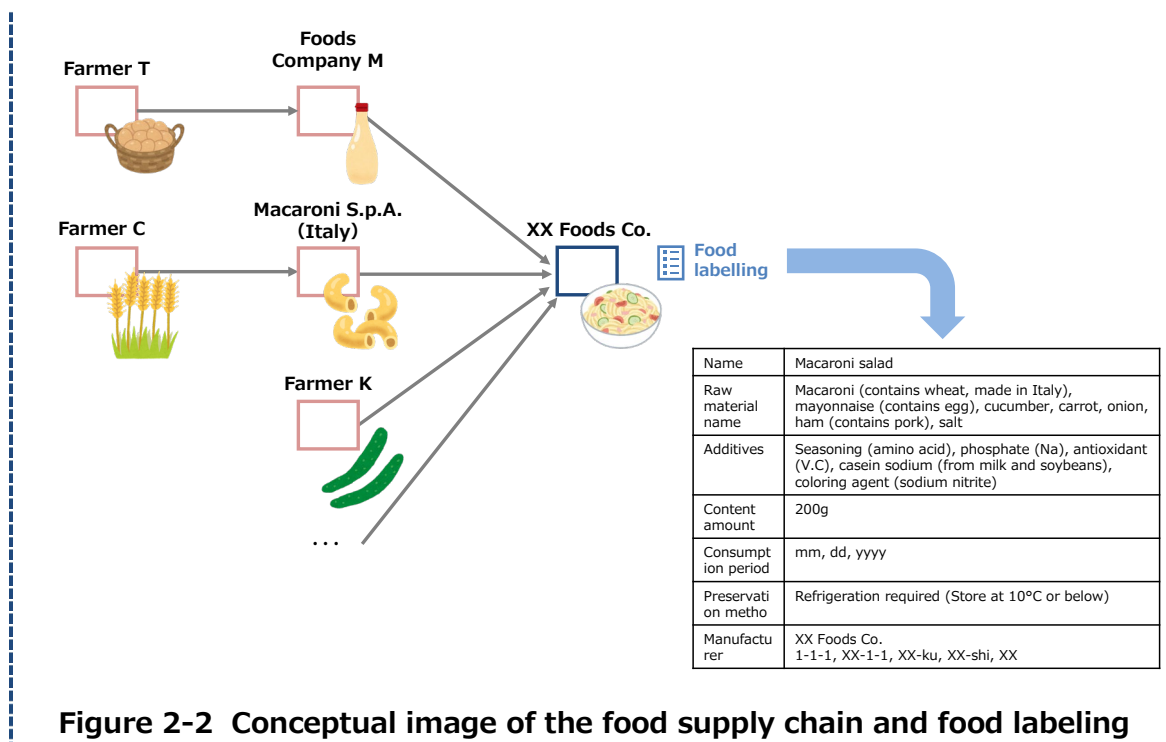
Table 2-1 Image of an SBOM in the scenario (matrix form)

ID	Supplier name	Component name	Version of the component	Other unique identifier	Dependency Relationship	Author of SBOM data	Timestamp
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in Application	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in Browser	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in Application	Company A	05-09-2022 13:00:00

The SBOM in Table 2-1, which is based on the simplified scenario above is only an image, and with this level of description, there may be no need to dare to manage it as an SBOM. However, the actual software is developed under a complex structure rather than a simple supply chain structure depicted in Figure 2-1. An actual SBOM will include not only proprietary software but also components developed by others, each of which will have complex implementation. Therefore, to improve software risk management and transparency in the software supply chain, it is important to use an SBOM to manage information about components in software, including their dependencies.

Column: Analogy between SBOM and food labeling

The SBOM is like the food label on food packaging. By reading food labels that visualize ingredients contained in food products, it is possible to prevent health hazards due to allergic accidents and to respond to food contraindications. Take macaroni salad as an example. As a result of manufacturing and processing, through the food supply chain, a food label is created as shown in Figure 2-2, which indicates ingredients. Like food labeling, an SBOM is a list of information regarding components contained in software, and the visualization of this information facilitates vulnerability response and risk management. Just as food labeling contributes to transparency in the food supply chain, SBOM contribute to transparency in the software supply chain. Note that, however, SBOM are more complex lists than food labeling as they include not only component names but also their versions and implementation. It shall also be noted that many SBOM are dynamically modified even after they are created, so it is important for SBOM users to manage them.



2.2. Benefits of SBOM

As shown in Table 2-2, there are three typical benefits of SBOM introduction: vulnerability management, license management, and increased development productivity. In addition to the direct benefits of vulnerability management, license management, and improved development productivity, each of these benefits has indirect benefits in product value and corporate value.

Table 2-2 Benefits of SBOM

Benefit		Item	Description
Vulnerability management	Direct benefits	Reduce residual vulnerability risks	Collecting vulnerability information and matching it with SBOM information to detect vulnerabilities can reduce the risk of residual vulnerabilities in software.
		Reduce vulnerability response time	SBOM tools can be used to detect new vulnerabilities in real-time and determine their impact, thereby shortening

Benefit		Item	Description
			the initial response time.
		Reduce cost of vulnerability management	Automated management using SBOM tools reduces management costs compared to manual management.
	Indirect benefit	Increase product/corporate value	Reduced vulnerabilities in products and faster vulnerability responses increase the value of the product and the company.
		Improve cyber hygiene	More products with fewer vulnerabilities improve the overall security of cyberspace. (Reducing the risk of attacks through steppingstone exploits.)
License management	Direct benefit	Reduce risks of license violations	Risks of license violations due to failure to identify OSS can be reduced.
		Reduce costs of license management	Compared to manual management, automated management using SBOM tools reduces administrative costs.
	Indirect benefit	Increase product/corporate values	Reduced risks of product license violations increase the value of the product and the company.
Development productivity	Direct benefits	Prevented development delays	Early identification of component problems prevents development delays.
		Reduce development costs	Early identification of component problems reduces response costs.
		Cut development time	When selecting components to be used, workloads related

Benefit		Item	Description
			to the selection are reduced by referring to past SBOM for similar products.
		Improving efficiency of compliance	Efficiency improvements in areas such as certification, laws and regulations compliance, and export control management.
		Improving motivation in the workplace	Improvements motivation in the workplace, as a result of increased efficiency and productivity.

Of the benefits of SBOM introduction, the most notable are the benefits of vulnerability management, i.e., the series of vulnerability response processes that detect, prioritize, fix, and mitigate software vulnerabilities. Most software in recent years has been developed under a complex supply chain structure that includes not only proprietary software developed in-house but also many components developed by other companies and the OSS community. These components often have complex hierarchical structures and implementation. For example, if a Java application uses Apache Log4j as a component, Log4j is positioned as a subordinate component and may be difficult to identify through normal component management. However, even a subordinate component may have security implications if the component contains vulnerabilities.

To reduce the residual risk of vulnerabilities, it is important to effectively implement continuous monitoring of vulnerabilities based on information about the components in use. In this regard, by implementing an SBOM and checking each component against the vulnerability database, it is possible to efficiently check for known vulnerabilities and, as a result, reduce the risk of residual vulnerabilities in the software. Also, when a new vulnerability is revealed for a certain component, if organizations do not manage their software using an SBOM, organizations may not know whether their software contains that component or not, and they may be affected by the vulnerability without knowing it. As shown in Figure 2-3, if an SBOM is in place when a vulnerability is revealed in a component, the impact of the vulnerability can be immediately recognized, and the time required to address the vulnerability can be reduced. In addition, by sharing information about

software with partner companies, organizations affected by vulnerabilities, and software users, the time required to address vulnerabilities can be shortened. It will also contribute to the understanding of the actual situation when components are rewritten or added illegally by third parties in the supply chain. Furthermore, by utilizing SBOM for inter-organizational sharing, the workloads required for sharing software information can be reduced.

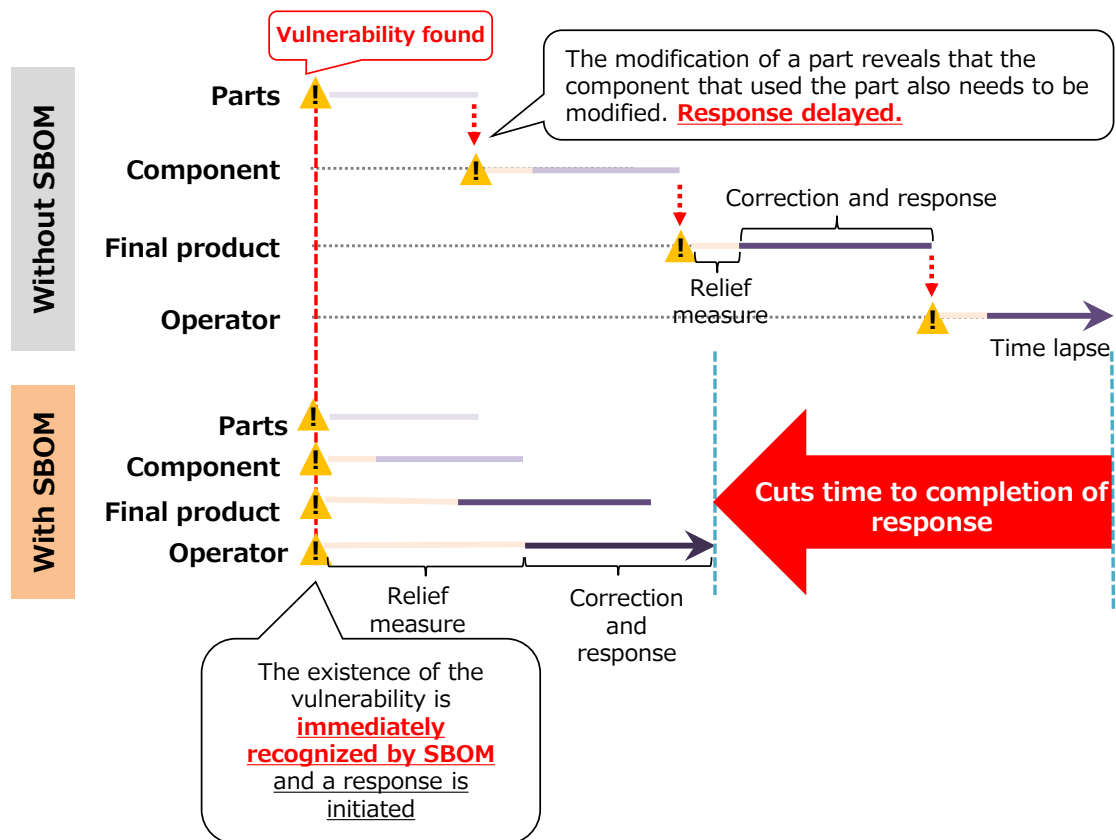


Figure 2-3 Benefits of reducing vulnerability response time by implementing an SBOM

Introducing an SBOM can reduce the cost of vulnerability management. Figure 2-4 shows the results of the cost evaluation of vulnerability management using SBOM in a medical device PoC conducted in FY2022. Here, it is assumed that the target software has about 80 components and the unit cost of workloads is ¥10,000 per hour. In the PoC, management by SBOM was performed using an SBOM tool. Manual component management requires manually identifying a list of components. It is necessary to manually search the vulnerability information database (e.g., NIST NVD) to check whether each component contains

vulnerabilities. When a vulnerability is revealed, it is necessary to check whether each component is affected or not by comparing the component information with the vulnerability information. On the other hand, SBOM management requires workloads to maintain the SBOM tool environment and to learn the SBOM tool. However, since the analysis and identification of components can be done automatically, the analysis and identification of components themselves require almost no workload. When a new vulnerability is revealed, it is automatically reflected in the SBOM tool, and it is possible to identify in real time whether components are affected. In such a case, although confirmation of the analysis/identification results is required, the cost of vulnerability management can be significantly reduced. In the PoC, it was confirmed that the workloads required for the SBOM were reduced to about 30% of those required for manual vulnerability management. It shall be noted, however, that the cost of commercial SBOM tools is incurred, but the more components are targeted, the more the cost will be divided proportionally.

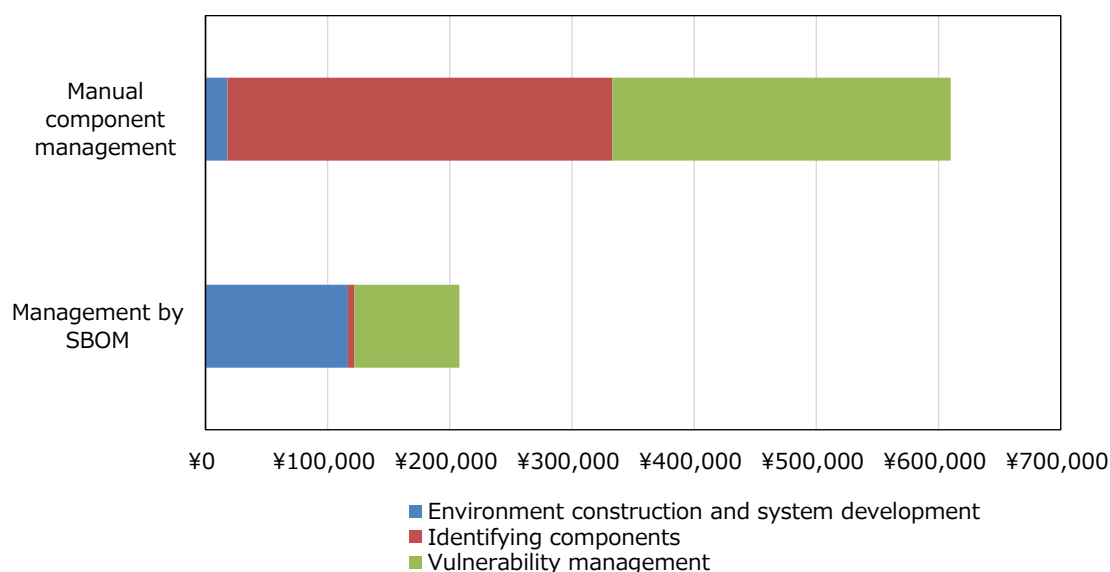


Figure 2-4 Results of reducing vulnerability management costs through the SBOM management (From the results of FY2022 PoC in the medical device sector)⁹

⁹ The cost of SBOM tools is not included. Also, for “vulnerability management,” vulnerability correction work and reporting work for which workloads do not vary significantly depending on whether or not SBOM is used, are excluded, taking into account workloads for vulnerability identification and risk assessment.

Indirect benefits of SBOM introduction for vulnerability management include increased product and corporate values resulting from reduced vulnerability risk, and in the big picture, there is also the benefit of increased security in cyberspace due to more products with fewer vulnerabilities.

The second benefit of SBOM introduction relates to license management. Specifically, benefits are found in the sequence of processes of identifying licenses for the components included in the software and handling them according to the requirements of each license. Most software in recent years includes OSS. Violation of OSS licenses can have major consequences, including suspension or recall of software sales, payment of fines, and damage to the company's brand image. Overseas, there have been several cases of lawsuits for violation of OSS licenses, including a case in which 14 companies, including consumer electronics manufacturers, were prosecuted for violation of the GNU General Public License (GPL) in 2009, a case in which a media player manufacturer was prosecuted for violation of the GPL in 2013, and a case in which a television manufacturer was prosecuted for violation of the GPL in 2021. When using OSS, it is necessary to take appropriate action according to the type of license, for example, in the case of GPL, GPL also applies to derivatives, and if new software is created and distributed by combining GPL with other software, the software must comply with the conditions imposed by GPL. In the case of the Mozilla Public License (MPL), the MPL is applied to derivative works as well as the GPL, but the MPL is not applied to new software created by combining them. Therefore, when using OSS, it is necessary to check all OSS licenses at one's own risk and comply with each license, but it is not easy to manage OSS license information without false negatives. By implementing an SBOM to manage components, including license information, the risk of license violations can be reduced, as well as the cost of license management as in vulnerability management. Furthermore, SBOM can protect the organization from financial risks arising from license violations, thus contributing to increased product and corporate values.

The third benefit of SBOM introduction is that it improves the software development life cycle (SDLC) and increases development productivity. When an SBOM is created in the early stages of software development, issues related to components, such as known vulnerabilities in the components or licensing issues, can be addressed in advance. Early identification of these problems can prevent development delays and reduce response costs. In addition, by managing information about components approved for use within the company, as an SBOM, it is no longer necessary to investigate and approve components each time they

are developed, and as a result, a reduction in development workloads can be expected. Regarding the benefits of increased development productivity, the Linux Foundation surveyed 412 global organizations in the third quarter of 2021¹⁰ and found that 51% of the responding organizations cited the benefits of SBOM as “making it easier for developers to understand the implementing between a wider range of complex projects. This is a higher percentage than the benefits of vulnerability management (49%) and license management (44%).

In this section, three typical benefits of SBOM introduction: benefits in vulnerability management, benefits in license management, and benefits in increased development productivity are mentioned, but other possible benefits exist. For example, management through SBOM can facilitate software EOL management. There are various benefits to be gained from introducing SBOM, but it is necessary to consider which benefits are particularly important based on the issues that your organization wants to solve by introducing SBOM and the purpose of introducing SBOM.

Column: Effect of SBOM on Log4j vulnerability (Log4Shell)

In December 2021, an arbitrary code execution vulnerability (commonly known as Log4Shell) was discovered in Apache Log4j, a log output library. The OSS Log4j is available free of charge and includes various functions, so it has been used for various purposes as a standard module for log output in Java systems. However, exploitation of discovered vulnerabilities and unauthorized access to applications running Log4j may lead to information leaks, malware infection, and other damage. In the “2021 Top Routinely Exploited Vulnerabilities” published by CISA and other organizations in the U.S.¹¹, Log4Shell ranked first, regardless of vulnerabilities discovered in December 2021, and the scope of this vulnerability's impact is immeasurable.

In addition to the fact that Log4Shell vulnerabilities are deployed in many software and are easy to attack, another reason for the widespread exploitation of Log4Shell vulnerabilities is that they are built in as components, so suppliers and software users are unaware of the impact of the vulnerabilities and no countermeasures are being implemented. Specifically, as shown in Figure 2-5, if Log4j components exist deeper than the range of components that software users can see (and are aware

¹⁰ See Footnote 2.

¹¹ CISA, Alert (AA22-117A) 2021 Top Routinely Exploited Vulnerabilities
<https://www.cisa.gov/uscert/ncas/alerts/aa22-117a>

of), then Log4j vulnerabilities can be exploited to affect software users, while software users are not aware of them.

By implementing an SBOM that includes multi-tier components, when a Log4j vulnerability is discovered, it is possible to immediately check whether the software in use is affected, thereby accelerating a response to the vulnerability. This reduces the risk of vulnerabilities being exploited and contributes to reducing the cost of responding to vulnerabilities and identifying the impacted area.

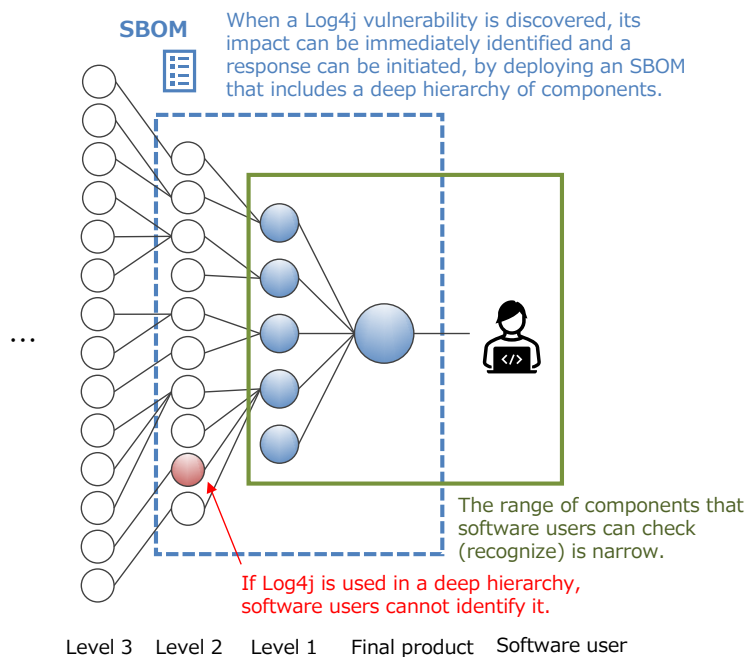


Figure 2-5 Image of software component hierarchy

2.3. "Minimum Elements" of SBOM

In response to a May 2021 U.S. Executive Order, NTIA released a document in July 2021 on the definition of "Minimum Elements" of the SBOM¹². The NTIA's definition of "Minimum Elements" not only specifies "Data Fields," which are categories of information to be included in the SBOM, but also "Automation Support" and "Practices and Processes" categories that organizations implementing an SBOM shall consider. The specific "Minimum Elements" categories and definitions are shown in Table 2-3.

¹² NTIA, The Minimum Elements For a Software Bill of Materials (SBOM)

<https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom>

Table 2-3 Definition of “Minimum Elements” of SBOM by the U.S. NTIA

Minimum Elements	Overview	Definition
Data Fields	Document baseline information about each component that should be tracked	<p>This baseline component information includes:</p> <ul style="list-style-type: none"> ● Supplier name ● Component name ● Version of the component ● Other unique identifiers ● Dependency relationship ● Author of SBOM data ● Timestamp
Automation Support	Support automation, including via automatic generation and machine-readability	SBOM data should be created and shared using machine-readable and interoperable formats. Currently, SPDX, CycloneDX, and SWID tags, which have been developed through international discussions, should be used.
Practices and Processes	Define the operations of SBOM requests, generation, and use	<p>Organizations utilizing SBOM shall establish operational procedures for the following items:</p> <ul style="list-style-type: none"> ● Frequency ● Depth¹³ ● Known unknown¹⁴ ● Distribution and delivery ● Access control ● Accommodation of mistakes¹⁵

¹³As shown in Figure 2-9, software components are often hierarchical, and the SBOM depth refers to the depth to which components in this hierarchical structure should be included in the SBOM.

¹⁴ If the dependencies of a complete component are unknown in the created SBOM, it means that the fact that it is unknown is made explicit. For example, clarification that the existence of the dependency is unknown, and clarification of the extent to which the component has not been identified.

¹⁵ The NTIA states that “while internal management of supply chain data may be a best practice, it is still evolving.” and also mentions that “In light of the absence of perfection,

In utilizing the SBOM, it is essential to collect information about components and establish a consistent data structure. For this reason, the inclusion of information for uniquely identifying a component subject to SBOM is positioned as a “minimum element” in the category of data fields. The definitions of specific data fields are shown in Table 2-4. In addition to information about the name, version, and other identifiers of the component subject to the SBOM, the data fields should include items related to the names of the supplier and the SBOM author of the component in question, the dependency of the component, and the timestamp.

Table 2-4 Data Fields to Be Included in the SBOM as “Minimum Elements”

Entry	Description
Supplier Name	The name of the entity that develops, defines, and identifies a component.
Component Name	Designation assigned to a unit of software defined by the original supplier.
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version.
Other Unique Identifiers	Other identifiers that are used to identify a component or serve as a look-up key for relevant databases.
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.
Timestamp	Record of the date and time of the SBOM data assembly.

2.4. SBOM formats (Examples)

As specified in the “Minimum Elements” of the SBOM, SBOM data should be created and shared using a machine-readable and interoperable format. The use of a common format will not only streamline management within an organization but also increase interoperability when sharing SBOM across organizations, thus

consumers of SBOM should be explicitly tolerant of the occasional incidental error. This will facilitate constant improvement of tools.”

contributing to transparency in the software supply chain. The following three formats are examples of SBOM format that can be used:

- (1) SPDX¹⁶
- (2) CycloneDX
- (3) SWID tag (Software Identification tag)

SPDX supports a wide range of software component types, including snippets, files, packages, containers, and OS distributions. In addition, it provides a list of identifiers for uniquely identifying a component's license information. There is also a Japanese-originated format called SPDX Lite, which includes only the minimum number of items required to meet the SBOM elements required by process management standards and other standards. SPDX Lite is excellent for simple SBOM creation and management and is also characterized by its abundance of specifications and other documents created in Japanese. CycloneDX is a format designed with security management in mind, which enables a description of not only information about the software in question but also information about the known vulnerabilities in the software and the exploitability of those vulnerabilities. Finally, for SWID tags, there is a feature that allows SBOM to be managed along the software life cycle.

In this section, reconsidering the simplified scenarios presented in Figure 2-1, an example is given for SBOM created by Company A in different SBOM formats.

(1) SPDX

SPDX is an SBOM format developed by a project under the Linux Foundation, which was standardized as ISO/IEC 5962:2021 in September 2021. SBOM in the SPDX format describe information about components created according to the SPDX Specification, licenses, copyrights, and so on. SPDX supports Tag-Value (txt), RDF, XLS, JSON, YAML, and XML formats. Refer to 10.3.3(1) of the Appendix for the structure of the SPDX format, usage examples/purposes, and features.

In the simple scenario described above, when Company A creates an SBOM using the SPDX format of the Tag-Value format, the SBOM shown in Figure 2-6 is created. Here, the color relationship indicates the correspondence relationship between the conceptual image of SBOM shown in Table 2-1 and the items in the SPDX format.

¹⁶ Until SPDX v2.3, the abbreviation SPDX was used to refer to *Software Package Data Exchange*, but in SPDX v3.0 published in April 2024, it was defined to mean *System Package Data Exchange*.

As shown in Table 2-5, the SPDX format items can be supported for each of the “Minimum Elements” in the SBOM.

ID	Supplier name	Component name	Version of the component	Other unique identifier	Dependency Relationship	Author of SBOM data	Timestamp
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in Application	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in Browser	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in Application	Company A	05-09-2022 13:00:00

▼ SBOM in SPDX format (Tag-value format)

```
SPDXVersion: SPDX-2.2
DataLicense: CC0-1.0
DocumentNamespace: http://www.spdx.org/spdxdocs/8f141b09-1138-4fc5-ae5b-fc10d9ac1eed
DocumentName: SBOM Example
SPDXID: SPDXRef-DOCUMENT
Creator: Organization: Company A
Created: 2022-05-09T13:00:00Z
Relationship: SPDXRef-DOCUMENT DESCRIBES SPDXRef-Application-v1.1

PackageName: Application
SPDXID: SPDXRef-Application-v1.1
PackageVersion: 1.1
PackageSupplier: Organization: Company A
PackageDownloadLocation: NOASSERTION
FilesAnalyzed: false
PackageChecksum: SHA1: 75068c26abbed3ad3980685bae21d7202d288317
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
ExternalRef: SECURITY cpe23Type cpe:2.3:a:company a:application:1.1:*:*:*:*:*
Relationship: SPDXRef-Application-v1.1 CONTAINS SPDXRef-Browser-v2.1
Relationship: SPDXRef-Application-v1.1 CONTAINS SPDXRef-Protocol-v2.2

(Omitted below)
```

Figure 2-6 Example of SBOM in SPDX Format (Tag-Value format) in the scenario

Table 2-5 SPDX Items Corresponding to SBOM “Minimum Elements”

Data Fields of SBOM “Minimum Element”	Corresponding SPDX item
Supplier Name	PackageSupplier
Component Name	PackageName
Version of the Component	PackageVersion
Other Unique Identifiers	Combination of DocumentNamespace and SPDXID, ExternalRef
Dependency Relationship	Relationship (DESCRIBES; Representation by CONTAINS)

Data Fields of SBOM “Minimum Element”	Corresponding SPDX item
SBOM author (Author of SBOM Data)	Author
Timestamp	Author

SPDX is a format developed to effectively handle information about OSS license compliance. and is characterized by its ability to express detailed information structured down to the file level. The target components are not limited to snippets and files but can be extended to packages, containers, and OS distributions. The format was developed with the intention of automated processing, and as mentioned above, it has been internationally standardized as ISO/IEC 5962:2021, which is also a major feature.

There is also a Japanese-originated format called SPDX Lite, which includes only the minimum number of items required to meet the SBOM elements required by process management standards and other standards. SPDX Lite is designed for organizations that manually create license information and transfer only necessary information when SPDX compliance license information is too large to operate. Developed by the License Information Subgroup of the OpenChain Japan Work Group, SPDX Lite is also part of the ISO/IEC 5962:2021 standard as a subset of SPDX. The SBOM in the SPDX Lite format describes information such as components, license, and copyright, and supports Tag-Value (txt), RDF, XLS, JSON, YAML, and XML formats. Refer to 10.3.3(1) of the Appendix for the structure of the SPDX Lite format, examples and purpose of use, and features.

In the simplified scenario described above, if Company A creates an SBOM using the SPDX Lite format in XLS format, the SBOM will be created as shown in Figure 2-7. In the case of the SPDX Lite format in XLS format, SBOM information can be described by including two sheets, “Creation Information” and “Package Information,” in a single XLS file. Here, the colors indicate the correspondence between the conceptual image of the SBOM shown in Table 2-1 and the items in the SPDX Lite format. As shown in Table 2-6, SPDX Lite format items can be supported for items other than the “Dependency Relationship” of the “Minimum Elements” of SBOM.

ID	Supplier name	Component name	Version of the component	Other unique identifier	Dependency Relationship	Author of SBOM data	Timestamp
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in Application	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in Browser	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in Application	Company A	05-09-2022 13:00:00



SBOM in SPDX Lite format (xls format)

Creation Information Sheet

SPDX Version	SPDX-2.2
Data License	CC0-1.0
SPDX Identifier	SPDXRef-DOCUMENT
Document Name	SBOM Example
SPDX Document Namespace	http://www.spdx.org/spdxdocs/8f141b09-1138-4fc5-aefb-fc10d9ac1eed
Creator	Company A
Created	05-09-2022 13:00:00

Package Information Sheet

Package Name	Package SPDX Identifier	Package Version	Package File Name	Package Supplier	Package Download Location	Files Analyzed	Package Homepage	Concluded License	Declared License	Comments on License	Copyright Text	Package Comment	External Reference field
Application	234	1.1	Omitted	Company A	omitted								
Browser	334	2.1		Company B									
Compression Engine	434	3.1		Mr. C									
Protocol	534	2.2		Community P									

Figure 2-7 Example of SBOM in SPDX Lite Format (XLS Format) in the scenario

Table 2-6 SPDX Lite Items Corresponding to “Minimum Elements” of SBOM.

Data Fields of SBOM “Minimum Element”	Corresponding SPDX Lite item
Supplier Name	PackageSupplier
Component Name	PackageName
Version of the Component	PackageVersion
Other Unique Identifiers	Combination of SPDX Identifier and SPDX Document Namespace, PackageSPDX Identifier
Dependency Relationship	—
Author of SBOM Data	Author
Timestamp	Created

SPDX Lite is a format that extracts only the minimum necessary items from SPDX, enabling SBOM management with an emphasis on operability. SPDX has many items that need to be described and are intended to be managed through automatic processing, while SPDX Lite has a limited number of items, so manual management is practically possible. However, it should be noted that SPDX Lite includes only the minimum necessary items, so for example, items related to “implementing” specified in the NTIA's “Minimum Elements” cannot be expressed. Since the number of items is limited, it may not meet the requirements of upstream organizations when sharing SBOM within the supply chain. Therefore, it is desirable to confirm with suppliers when deciding whether to use SPDX Lite. Furthermore, it should be noted that manual management of SPDX Lite formatted SBOM may require more management workloads than automatic management.

(2) CycloneDX

CycloneDX is an SBOM format developed by an OWASP community project with the goal of developing a security focused SBOM format standard. The CycloneDX SBOM format includes information about components, licenses, and copyrights. CycloneDX supports JSON, XML, and Protocol Buffers (protobuf) formats. Refer to 10.3.3(3) of the Appendix for the structure, usage examples, and features of the CycloneDX format.

In the simplified scenario described above, if Company A creates an SBOM using the CycloneDX format in XML format, the SBOM will be created as shown in Figure

2-8. Here, the color relationships indicate the correspondence between the conceptual image of SBOM shown in Table 2-1 and the items in the CycloneDX format. As shown in Table 2-7, the items in the CycloneDX format can be made to correspond to each of the “Minimum Elements” for SBOM.

ID	Supplier name	Component name	Version of the component	Other unique identifier	Dependency Relationship	Author of SBOM data	Timestamp
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in Application	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in Browser	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in Application	Company A	05-09-2022 13:00:00

SBOM in CycloneDX format (XML format)

```
<?xml version="1.0" encoding="utf-8"?>
<bom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
serialNumber="urn:uuid:3e671687-395b-41f5-a30f-a58921a69b71" version="1"
xmlns="http://cyclonedx.org/schema/bom/1.3">
  <metadata>
    <timestamp>2022-05-09T13:00:00Z</timestamp>
    <authors>
      <author>
        <name>Company A</name>
      </author>
    </authors>
    <component type="application">
      <name>Application</name>
      <version>1.1</version>
      <hashes>
        <hash alg="SHA-1">75068c26abbed3ad3980685bae21d7202d288317</hash>
      </hashes>
      <cpe>cpe:2.3:a:company_a:application:1.1:*:*:*:*:*</cpe>
      <externalReferences />
      <components />
    </component>
    <manufacture>
      <name>Company A</name>
    </manufacture>
    <supplier>
      <name>Company A</name>
    </supplier>
  </metadata>

  (Omitted)

  <dependencies>
    <dependency ref="pkg:maven/org.company_b/browser@2.1">
      <dependency ref="pkg:maven/org.c/CompressionEng@3.1" />
    </dependency>
    <dependency ref="pkg:maven/org.community_p/protocol@2.2" />
  </dependencies>

  (Omitted below)
```

Figure 2-8 Example of SBOM in CycloneDX Format (xml format) in the scenario

Table 2-7 CycloneDX Items Corresponding to SBOM “Minimum Elements”

Data Fields of SBOM “Minimum Element”	Corresponding CycloneDX item
Supplier Name	component/supplier/name
Component Name	component/name

Data Fields of SBOM “Minimum Element”	Corresponding CycloneDX item
Version of the Component	component/version
Other Unique Identifiers	serialNumber, component/cpe
Dependency Relationship	implementing/dependency ref
Author of SBOM Data	metadata/authors/author/name
Timestamp	metadata/timestamp

One of the features of CycloneDX is that it is an SBOM format with security management in mind. CycloneDX Version 1.4, released in January 2022, adds “Vulnerabilities” to the object model, allowing the description of known vulnerabilities in third-party software and OSS included in the SBOM and the potential for exploitation of those vulnerabilities. CycloneDX, like SPDX, is also a format intended for automatic processing by tools.

(3) SWID tag (Software Identification tag)

SWID tags were developed for the purpose of tracking software installed on devices managed by an organization. SWID Tags were defined by ISO in 2012 and updated in 2015 as ISO/IEC 19770-2:2015. With a SWID tag, as part of the software installation process along the software lifecycle, when software is installed on a device, information about the installed software, called a tag, is assigned to the device, and when the software is uninstalled, the tag is removed. An SBOM in the SWID tag format describes information such as software installed on the device and patches applied to the software created according to the SWID tag. SWID tag supports XML format. SWID tag defines tags that indicate information about software installed in a device to understand the life cycle of the target device. Each tag can present information such as the author of the tag, the software installed on the device, and the dependencies by linking to other software, and can be used as an SBOM of the target device. Refer to 7.3.3 (4) of the Appendix for more information about the structure of the format, examples of use and purpose of use, and characteristics of the SWID tag.

In the simplified scenario described above, if Company A creates an SBOM using a SWID tag in XML format, the SBOM will be created as shown in Figure 2-9. In this figure, the color relationship shows the correspondence between the conceptual image of the SBOM shown in Table 2-1 and the items in the SWID tag format. As shown in Table 2-8, an item in the SWID tag format can be made to correspond

to each item of the SBOM “Minimum Elements”.

ID	Supplier name	Component name	Version of the component	Other unique identifier	Dependency Relationship	Author of SBOM data	Timestamp
1	Company A	Application	1.1	234	Primary	Company A	05-09-2022 13:00:00
2	Company B	Browser	2.1	334	Included in Application	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in Browser	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in Application	Company A	05-09-2022 13:00:00



SBOM in SWID Tag format (XML format)

```
<SoftwareIdentity
  xmlns="http://standards.iso.org/iso/19770/-2/2015/schema.xsd"
  xmlns:swid="http://www.w3.org/2001/04/xmlenc#sha512"
  name="application"
  tagId="Company A/application@1.1"
  version="1.1">
  <Entity name="Company A" role="tagCreatorsoftwareCreator" />
  <Meta title="Company A Application v1.1" timestamp="2022-05-09T13:00:00Z" />
  <Link href="swid:Company B/browser@2.1" rel="component" />
  <Link href="swid:Community P/ptotocol@2.2" rel="component" />
  <Payload >
    <File name="Company-A-application-1.1.exe"
    sha512:hash="BC55DEF84538898754536AE47CC907387B8F61D9ACD7D3FB8B8A624199682C8FBE6D163108
    8AE6A322CDDC4252D3564655CB234D3818962B0B75C35504D55689"/>
  </Payload>
</SoftwareIdentity>

(Omitted below)
```

Figure 2-9 Example of SBOM in SWID Tag Format (xml Format) in the scenario

Table 2-8 SWID Tag Entry Corresponding to SBOM “Minimum Elements”

Data Fields of SBOM “Minimum Element”	Corresponding SWID tag item
Supplier Name	<Entity> @role(tagAuthor) @name
Component Name	<SoftwareIdentity> @name
Version of the Component	<SoftwareIdentity> @version
Other Unique Identifiers	<SoftwareIdentity>@tagId
Dependency Relationship	<Link> @rel @href
Author of SBOM Data	<Entity> @role(softwareAuthor) @name
Timestamp	<Meta> @timestamp

The SWID tag is a format related to software identification. It is also a format that can include information related to security, such as information about component licenses, information about patches and updates, and information about

vulnerabilities and threats.

So far, examples of SBOM in SPDX, SPDX Lite, CycloneDX, and SWID tags have been shown. Many formats are intended for automatic processing and management using SBOM tools. SBOM tools can be used to automatically create SBOM by scanning software source codes and binary files and automatically detecting components contained in the software. In addition, some SBOM tools can streamline administrative tasks by providing continuous access to vulnerability and license information. Therefore, it is practical for organizations implementing an SBOM to create and manage the SBOM using SBOM tools. Typical SBOM tools, not only commercial SBOM tools but also OSS SBOM tools, are shown in 7.3.2 of the Appendix.

Organizations that implement SBOM should evaluate and select multiple SBOM tools based on their own objectives for implementing an SBOM and the scope of application of SBOM, after clarifying the viewpoints for selecting SBOM tools. Refer to the points to be implemented and recognized in the selection of SBOM tools.

When SBOM tools are used to manage SBOM, SBOM documents in Tag-Value or XML formats, as shown in Figure 2-6 through Figure 2-9, can be created and managed without much consideration. Many commercial SBOM tools have several dashboard functions, which enable easily displaying the list of components included in an SBOM, as well as listing and graphing information about vulnerabilities and license compliance. of each component.

2.5. Myths and facts

Despite the advantages of SBOM introduction, the penetration rate of SBOM in Japan is not high. There are various possible reasons for this, including the cost of SBOM introduction, technical issues, and human resource issues, but there are also other issues such as the lack of proper recognition of the effectiveness and positioning of SBOM. In response to these challenges, the US NTIA released a document titled “SBOM Myths vs. Facts” in 2021¹⁷ to clarify misconceptions and facts about SBOM. Below is a summary of the misconceptions and facts presented in the NTIA document.

Myth: SBOM are a roadmap to the attacker

¹⁷ NTIA, SBOM Myths vs. Facts

https://www.ntia.gov/files/ntia/publications/sbom_myths_vs_facts_nov2021.pdf

[Fact] Attackers can leverage the information contained in SBOM. However, the defensive benefits of transparency far outweigh this common concern as SBOM serve as a “roadmap for the defender”. For attackers, SBOM and software transparency information are of limited effectiveness, and attackers generally do not need SBOM. For example, the WannaCry ransomware attack does not require SBOM as a prerequisite for the attack.

Myth: An SBOM alone provides no useful or actionable information

[Fact] The baseline component information supports a number of use cases for those who produce, choose, and operate software. For example, during an active attack, an SBOM allows an enterprise to answer, “Am I affected?” and “Where am I affected?” in minutes or hours, instead of days or weeks. Additionally, the baseline component information enables vital transparency and auditability, allowing for further expansion and enrichment in additional use cases.

Myth: An SBOM needs to be made public

[Fact] An SBOM does not need to be made public. The act of making an SBOM is separate from sharing it with those who can use this data constructively. The author may advertise and share the SBOM at their discretion. In other cases, sector-specific regulations or legal requirements may require more or less access to the SBOM.

Myth: An SBOM will expose my intellectual property/trade secrets

[Fact] SBOM are a summary of included software components and do not expose intellectual property (IP). Patents and algorithms are not included. n SBOM is just a “list of ingredients”, not a “recipe” like a patent or an algorithm. In addition, SBOM does not include the software source code itself¹⁸. It is important to note that the intellectual property of third-party developed components, such as patents and algorithms, belong to the component developer or copyright holder.

Myth: No processes exist to support scalable production and use of SBOM

[Fact] Software composition analysis tools have been used internally in some sectors for more than a decade. Regarding software transparency, NTIA activities, executive orders, standardization of the SBOM format, and other

¹⁸ Although SBOM does not include trade secrets such as software source code, it is necessary to be aware that it may include other proprietary information such as information from software providers and vendors, so appropriate management is required.

activities are progressing. In some industries, software transparency has been under discussion, and PoC for more than 5 years support the adoption of SBOM formats.

In addition, through PoC and other activities conducted in Japan in FY2022, further specific myths and facts have been made clear.

Myth: Only the components directly used by the target software should be subject to SBOM management

[Fact] Vulnerability management may be insufficient if the components recursively used by direct components are excluded. Discussions by experts are ongoing regarding the “depth” of SBOM (i.e., up to what level of components should be included in SBOM).

Myth: No special consideration is needed to select SBOM tools

[Fact] Regarding tools to support SBOM production, several commercial tools and OSS tools provided as OSS are already available. By using OSS tools, the tools themselves can be obtained at no cost, but compared to commercial tools, the manuals and support for introduction and utilization are often limited, which may result in significant costs incurred in learning how to use the tools. In addition, compared to commercial tools, the scope of support and performance are usually limited, and there is a possibility that the purpose of SBOM implementation cannot be achieved. It is necessary to select tools based on the objectives of the company's SBOM implementation.

Myth: SBOM tools can be utilized to fully identify the components contained in the target software

[Fact] Although SBOM tools can be used to efficiently create SBOM, there may be cases where false positives or false negatives in the production of SBOM, making it impossible to create accurate SBOM. Therefore, it is important to consider other ways to reaffirm the accuracy of the SBOM (for example, reviewing the SBOM created by the tool). In addition, libraries that are dynamically added at runtime, such as runtime libraries, cannot be identified because SBOM tools do not analyze the substance of the library. In such cases, it is necessary to prepare separate configuration information and execution environment for the library by using other tools such as a package manager, and having the SBOM tool recognize them so that recursive components can be identified.

Myth: There is a need to respond to all vulnerabilities output by SBOM

tools

[Fact] It is necessary to prioritize vulnerabilities when responding to risks based on the output. Prioritization should occur based on the impact of the vulnerability, the results of the risk assessment, and the cost of responding to the vulnerability. In doing so, it should be noted that not all vulnerabilities are available for use, and some vulnerabilities that exist are not affected. In the case of manual SBOM management, it is necessary to manually identify the existence of vulnerabilities by using the vulnerability database, evaluate each vulnerability individually, and consider the response policy for each vulnerability, which may require significant management costs.

Myth: Granularity of the SBOM components should be standardized throughout the supply chain and only the necessary component information should be retained

[Fact] Currently, the granularity of “affected software” in vulnerability information databases such as Japan JVN and U.S. NVD is not systematized, and limiting the granularity of components may lead to false negatives in identifying vulnerabilities. Therefore, it is an effective practice to retain component information not only for OSS but also for in-house products.

Myth: SBOM only covers packaged and embedded software

[Fact] Not only software but also IT systems can be covered by an SBOM. In addition, SBOM for online applications such as SBOM for container images, SBOM for SaaS software, and SBOM for cloud services are also being discussed mainly in the U.S, but challenges specific to the cloud environment SBOM are also mentioned.

Myth: Only three formats of SBOM are allowed: SPDX, CycloneDX, and SWID tags; SBOM based on proprietary formats are not allowed

[Fact] According to the definition of the U.S. NTIA, an SBOM is “a machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships.” Even proprietary formats can be considered SBOM if they meet this definition. However, as stated in Section 2.2, since the “automation support” is positioned as the “Minimum Elements” of SBOM, and since automated processing improves efficiency, it is desirable to consider adopting an automatically processable format whenever possible.

3. Basic guidance and overall view on SBOM introduction

3.1. Basic guidance for SBOM introduction

Prior to introducing SBOM, it is necessary to determine the scope of software for which to create SBOM, as well as to clarify the issues that one's own organization wishes to solve by implementing SBOM and the purpose of the introduction. For example, for a large-scale product with a huge number of components, if the purpose is to create and share SBOM that include component dependencies, then it is expected that SBOM will be created and managed using commercial SBOM tools. Also, for small-scale products that do not have many components, if the purpose is to manually manage the version of the components only for the minimum items, an SBOM may be created using the SPDX Lite format. Depending on the purpose of the SBOM introduction, the scope of application of the SBOM, such as the items, format, creation range, and sharing range of an SBOM to be created will vary to a larger extent. An organization considering implementing an SBOM should first identify its own software management issues that it intends to solve by implementing SBOM and clarify the purpose of the introduction, before creating, operating, and managing the SBOM.

3.2. SBOM introduction process

The process related to SBOM introduction can be divided into three main phases. Specifically, there are three phases: the environment construction and system development phase related to SBOM introduction, the SBOM creation and sharing phase, and the SBOM operation and management phase. Figure 3-1 shows the main items to be implemented and an overview of the implementation in each phase.

In the environment construction and system development phase, the scope of SBOM introduction will be clarified, and an environment and system for SBOM creation and sharing will be established. In the SBOM creation and sharing phase, the SBOM is created and, if necessary, shared with external parties. SBOM is a method of software management. How to manage software by using an SBOM is particularly important. Therefore, as part of the SBOM operation and management phase, vulnerability management and license management need to be conducted based on SBOM information, and the SBOM itself should be managed appropriately.

The following sections show the main implementation items for each phase and the points to note when introducing an SBOM.

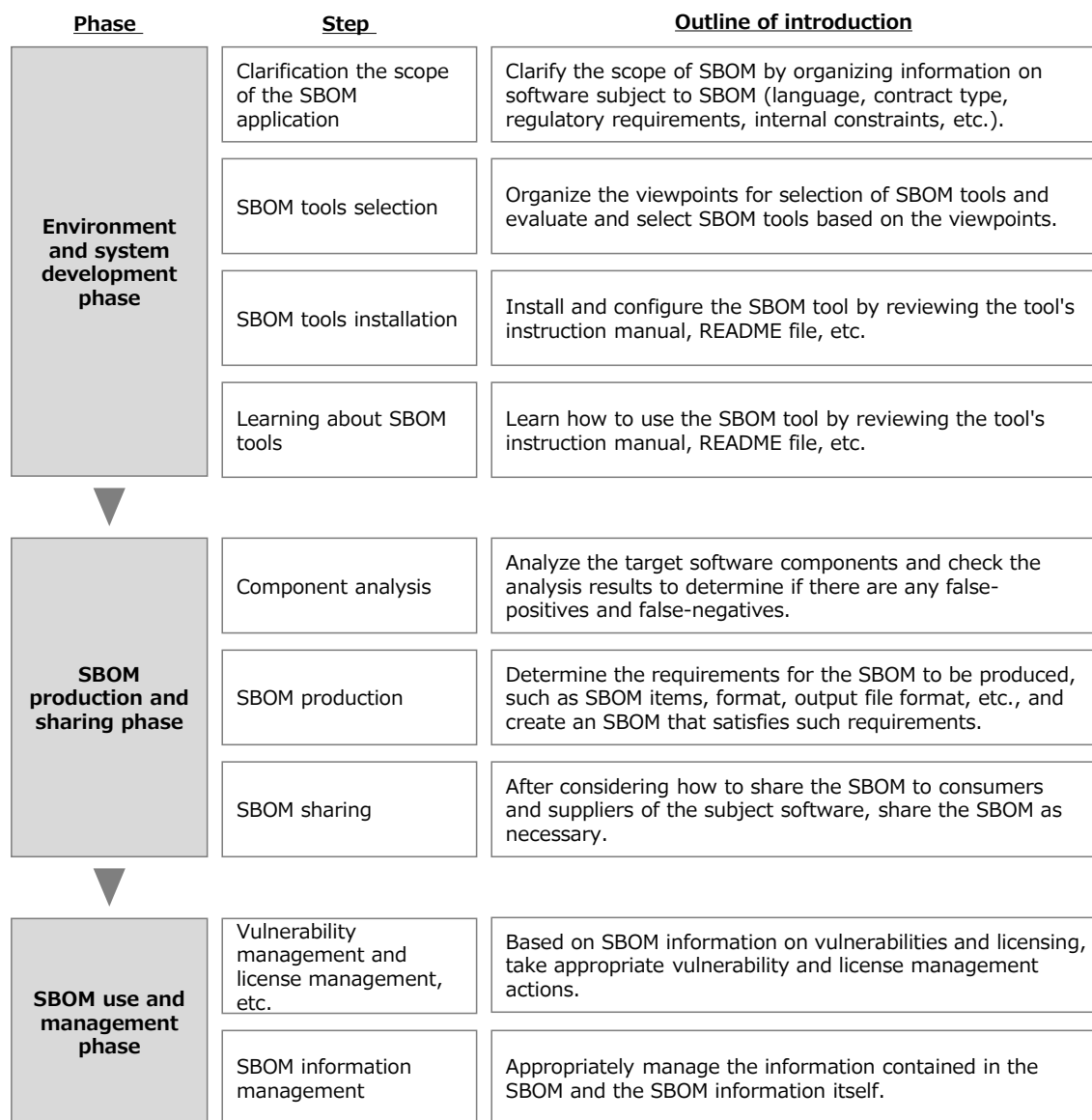


Figure 3-1 SBOM introduction process

4. Environment and system development phase

To introduce an SBOM, it is first necessary to establish an environment and a system related to the SBOM. This section presents the items that SBOM-introducing organizations should implement and the points that they should be aware of in the environment construction and system development phase.

4.1. Clarification the scope of the SBOM application

[Actions for the introduction of SBOM]

- ☐ Clarify information about the target software, such as information about development language, component type, development tools, etc.
- ☐ Create an accurate configuration diagram of the target software and visualize the target of the SBOM application.
- ☐ Clarify the contractual form and business practices with users and suppliers of the subject software.
- ☐ Confirm regulations and requirements for SBOM regarding the target software.
- ☐ Clarify the constraints within the organization (e.g., system constraints, cost constraints) regarding the introduction of SBOM.
- ☐ Clarify the scope of the SBOM application 5W1H (Five Ws and How) based on the organized information.

[Points to be aware of for SBOM introduction]

- By utilizing the knowledge of developers inside and outside the organization, it is possible to efficiently collect information about the target software.
- The scope of risk management can be clarified by creating an accurate configuration diagram of the target software and by visualizing the target of the SBOM application.

An organization introducing an SBOM needs to clarify the scope of application of SBOM based on their own issues to be solved by SBOM introduction and the

purpose of SBOM introduction. The scope of the SBOM application can be classified into the Five Ws and How (5W1H) perspectives shown in Table 4-1. There are multiple application items (options) in each perspective.

Table 4-1 Scope of SBOM application (Five Ws and How)

Perspective	Main application item (option)
Organization producing an SBOM (Who)	<ul style="list-style-type: none"> • Produced internally • Produced by suppliers with business contract • Produced by suppliers without business contract (e.g., OSS community)
Timing of producing an SBOM (When)	<ul style="list-style-type: none"> • During product planning or development planning • During program development • During software built • At software delivery • At component upgrading
Entity to use the SBOM (Who)	<ul style="list-style-type: none"> • Software user • End-product vendor • Development vendor • End-product user
Scope of components covered by the SBOM (What, Where)	<ul style="list-style-type: none"> • Only components directly used by the development entity • Components that are recursively used from components without a development consignment contract such as off-the-shelf products
Means of producing the SBOM (How)	<ul style="list-style-type: none"> • Producing an SBOM manually based on configuration management information • Producing an SBOM automatically using SBOM tools • Producing part of an SBOM manually based on configuration management information and the other part of the SBOM automatically by using SBOM tools

Perspective	Main application item (option)
Scope of utilizing the SBOM (Why)	<ul style="list-style-type: none"> • Vulnerability management • License management • Improvement in development productivity • Asset management and traceability • Sharing information about components to users and/or suppliers
SBOM formats and items (What)	<ul style="list-style-type: none"> • Standard formats (SPDX, SPDX Lite, CycloneDX, SWID tag, SPDX Lite) • Data Field of the "Minimum Elements" • Proprietary formats used as regulations/requirements or industry practice

The extent of the SBOM coverage is determined by the combination of these applicable items. It should be noted that the cost of implementing an SBOM will vary depending on which applicable items are selected. In addition, there is a possibility of selecting multiple applicable items for a single perspective. To determine the applicable items, it is necessary to organize information about the target software of SBOM and internal restrictions on SBOM introduction.

For the target software, it is desirable to first organize information about the following¹⁹.

- **Software language**

Example: Python, Java, Go, JavaScript, Rust, Swift, Objective-C, C, C++, VisualBasic

- **Form of component**

Example: Libraries, applications, middleware, database services

- **Development environment tool**

Example: Visual Studio, Eclipse, Android Studio, Xcode

- **Build tool**

Example: Jenkins, Circle CI, GitHub Actions, Gradle, Maven

- **Configuration management tool**

Example: GitHub, Gitlab, Team Foundation Server, Ansible

- **Data formats handled by the organization**

¹⁹ Note that the examples for each item are not exhaustive and are not limited to the content of the examples.

Example: Source codes, packages, containers, binary data

- **Operating environment**

Example: OS, CPU architecture

In organizing such information, it is effective to utilize the knowledge of developers inside and outside the organization. When SBOM tools are used to create SBOM, it is necessary to understand at least the development language and the form of components, since each tool supports different languages and different component forms. To clarify the scope of components to be covered by SBOM, it is desirable to visualize the composition of the target software. Specifically, it is desirable to create a diagram that visualizes the scope of the target software developed by the organization, the scope developed by suppliers with business contracts, and the scope developed by suppliers without business contracts (e.g., OSS). As an example, the following configuration diagram was created for the dental CT targeted in the PoC in FY2022. Based on this diagram, the scope of risk management has been clarified.

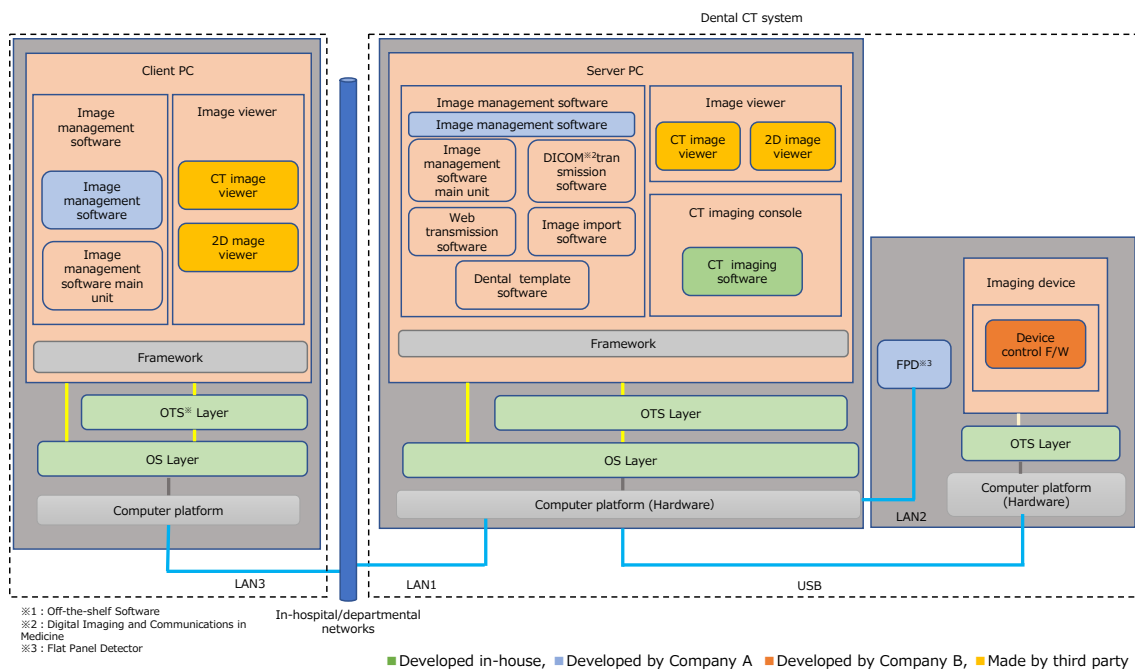


Figure 4-1 Example of the system configuration diagram of dental CT

In addition, to clarify the scope of components to be covered by the SBOM, it is desirable to organize types of contract forms and transaction practices with users and suppliers of the subject software. Specifically, it is desirable to organize information about the following items for each user and supplier of the subject

software.

- **Type of contract:** Development outsourcing, product sales, etc.
- **Provision of component information:** Not provided, provided without charge, available upon request, etc.
- **Declaration of third-party components:** Declaration for all OSS, declaration for some OSS based on license, etc.
- **Vulnerability notice:** Notification only for vulnerabilities that have been determined to be fixed, etc.
- **Vulnerability fix:** Only fix for vulnerabilities that have been determined to be fixed, etc.
- **Delivery form:** Binary package, embedded in equipment, license information (e.g., SaaS), executable module, etc.
- **Liability for damages**
- **Attribution of intellectual property rights:** Belongs to the company, belongs to the supplier, belongs to the supplier, etc.
- **Modification:** Software provided by a third party being used as is, modified by the company, etc.

Among the SBOM applicable items, it is desirable to confirm and organize the regulations and requirements for SBOM for the target software, to determine the format and items of SBOM and the scope of SBOM utilization. Currently, the number of software vendors that are required to provide SBOM is limited, but in the U.S., for example, software vendors that are subject to government procurement are encouraged to provide SBOM²⁰. In the EU, the Cyber Resilience Act, drafted in September 2022, includes SBOM requirements for digital products to be placed on the EU market²¹. In the medical device segment, the Medical Device Cybersecurity Guide, issued by the International Medical Device Regulators Forum (IMDRF), will be incorporated into the medical device regulations under the pharmaceutical affairs law²² and will be fully operational by the end of 2023. There

²⁰ Office of Management and Budget, Enhancing the Security of the Software Supply Chain through Secure Software Development Practices <https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf>

²¹ European Commission, Cyber Resilience Act <https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act>

²² Act on Securing Quality, Efficacy and Safety of Products Including Pharmaceuticals and Medical Devices

is a possibility that SBOM will be required in regulations in the future. Regulations and requirements may specify formats and items of the SBOM and the scope of SBOM utilization. Considering these circumstances, it is desirable to collect information about regulations and requirements related to the target software as needed and to clarify specific requirements when needed.

In considering items to be applied to SBOM, it is, of course, necessary to consider the constraints within the organization for SBOM introduction. The most likely constraints are those related to the organizational structure and costs. If these constraints are severe, there is a possibility that only limited SBOM application items can be selected. It is then desirable to confirm and organize the constraints within the organization in advance to organize the scope of the SBOM application.

Based on the organized information, it is desirable to consider and clarify the applicable items for each of the above-mentioned Five Ws and How (5W1H) aspects of the scope of the SBOM application. It should be noted that the scope of the SBOM application varies depending on the scope and level of risk to be addressed. For example, if an SBOM is to be created for medical devices that will be required by regulations and requirements in the future, it is assumed that not only the organization itself but also suppliers with whom it has business contracts will create an SBOM at the time of software build and that the SBOM will be used by medical institutions as users. The scope of components to be covered by SBOM is not limited to components used directly but also includes components used recursively, and it is expected that SBOM tools will be used to automatically create an SBOM for vulnerability management and license management. As for the format and items of SBOM, it is desirable to create an SBOM based on a format that can be processed automatically, such as SPDX, and that includes the items required by the regulations.

4.2. SBOM tools selection

[Actions for the introduction of SBOM]

- ☐ Organize the viewpoints for the selection of SBOM tools considering the development language of the target software and the constraints within the organization.
(Examples of selection viewpoints: functions, performance, analyzable information, analyzable data format, cost, supported formats, component

analysis method, support systems, coordination with other tools, form of provision, user interface, operation method, supported software languages, Japanese support, etc.)

- ☐ Evaluate and select multiple SBOM tools based on the organized viewpoints.

[Points to be aware of for SBOM introduction]

- Since the use of multiple SBOM tools can be inefficient, it is advisable to consider whether the minimum number of SBOM tools should be used for a given purpose.
- Commercial SBOM tools are generally expensive. On the other hand, OSS SBOM tools may require many workloads for implementation and operation due to the lack of information about environmental maintenance and learning.
- Compared to commercial SBOM tools, OSS SBOM tools often have limited functions and performance: recursive use components cannot be detected, there are limitations on readable SBOM formats, license false negatives occur, or the installation environment is limited.
- For on-premises SBOM tools, the installation environment may be restricted. In addition, with SaaS-type SBOM tools, it is necessary to confirm that the tool is not structured to transmit sensitive source code information to external parties.
- It is necessary to select SBOM tools that can be easily integrated into the existing development process and to operate them in a way that does not place a burden on developers so that the implementation of SBOM does not cause a significant reduction in development efficiency.
- In selecting an SBOM tool, it is effective to experience the actual use of the tool by using a free trial. If organizations find it difficult to set up and select a viewpoint, they may consult with a distributor who handles multiple SBOM tools and compare and evaluate the features, advantages, and disadvantages of each tool before selecting one.

An organization planning to implement an SBOM should build an environment and establish a system for creating SBOM corresponding to the clarified scope of the SBOM application. SBOM tools provide the most important facility for creating and

managing SBOM. When creating and managing an SBOM, SBOM tools are not necessarily essential. Formats such as SPDX Lite that can create and manage SBOM manually are also available. It is demonstrated, however, that in addition to reducing the workloads required for component management, the use of SBOM tools can efficiently enable the detection of dependencies among OSS and reuse of OSS, thereby reducing a lead time between announcement and identification of vulnerability. Therefore, it is realistic to use SBOM tools to create and manage SBOM, and this Guidance also assumes the use of SBOM tools.

Some SBOM tools are shown in 10.3.2 of the Appendix. SBOM tools are broadly divided into commercial tools and OSS tools. Commercial SBOM tools are generally expensive, but they have a rich user interface that enables intuitive SBOM creation and management. They have the advantage that the user can consult with vendors and distributors. Furthermore, there are SBOM tools that can be linked with various development tools and communication tools. Meanwhile, OSS SBOM tools often lack information for environmental maintenance and learning. Therefore, OSS tools may require many workloads to implement and operate and to investigate and respond to the cause when an error occurs. Also, compared to commercial SBOM tools, the functions and performance of OSS SBOM tools are often limited. For example, reused components cannot be detected; there is a limit to SBOM formats that can be read; sometimes licenses are not detected; and the environment is limited for SBOM introduction. Nevertheless, it should be noted that OSS SBOM tools are actively developed mainly by the OSS community, and their functionality and performance will be improved. In addition, some companies provide support services for OSS SBOM tools, and users of OSS SBOM tools are expected to receive support as needed.

While various commercial and OSS SBOM tools are available, it is desirable to organize the viewpoints of selection considering the development language of the target software and restrictions within the organization and to evaluate and select SBOM tools based on this viewpoint. Examples of possible selection viewpoints include those shown in Table 4-2.

Table 4-2 Viewpoints for selecting SBOM tools

Viewpoints	Description
Functions	SBOM tools have the following functions: component analysis, automatic matching of vulnerability and license information, risk quantification, visualization of dependencies and vulnerability information, automatic tracking of vulnerability and license information, alert function when a new vulnerability is detected, automatic reporting of advisory information, and import function of SBOM data. Since each SBOM tool supports different functions, it is advisable to sort out which functions are necessary based on the purpose of the SBOM introduction and the scope of the SBOM application.
Performance	In the detection of OSS and the matching of vulnerability and license information, the degree of false positives and false negatives is an important indicator. In addition, it is also important to determine how quickly new vulnerabilities are reflected in the tool when they are found. It is desirable to clarify what level of performance ²³ is required, based on the purpose of the SBOM introduction and the scope of the SBOM application.
Analyzable information	Information about components that can be analyzed varies, depending on the SBOM tool. Many commercial tools can automatically analyze the vulnerability and license information of components, and there are also tools specialized in analyzing vulnerability information and license information. It is desirable to sort out which information is necessary based on the purpose of the SBOM introduction and the scope of the SBOM application.
Analyzable data format	SBOM tools have conditions on the data formats that can be read during component analysis. It is desirable to determine data formats to be used for analysis, such as file format (compatibility by extension), type of supported package manager, and OS/CPU architectures on which the software can run.

²³ Methods for understanding tool performance include the following: using a free trial, loading the actual software to be analyzed and the SBOM, etc. to be used into the tool, and confirm that the tool can output accurate information; and checking with the developer or distributor of the tool vendor's database for specifications such as the number of OSS and vulnerabilities included in the database, the source of vulnerability information (JVN, NVD, etc.), and the update frequency of the database.

Viewpoints	Description
Cost	<p>In the case of commercial SBOM tools, a tool license fee is required. The fee structure differs depending on the tool, but many tools are offered on an annual subscription basis. Some tools offer multiple OSS analysis methods as an option, and others offer a plan that enables various consultations on OSS management, not only for responding to inquiries. the method for calculating license fees varies by tool, based on factors such as the number of developers, the scale of the organization, and the amount of analysis code. There may be economies of scale when the entire organization adopts the tool even if it is expensive. It is desirable to determine how much cost can be spent on SBOM tools, considering the cost constraints within the company.</p>
Supported formats	<p>Depending on the SBOM tool, it is possible to import/create SBOM only in a specific SBOM format (SPDX, SPDX Lite, CycloneDX, SWID tags, etc.). As for SBOM creation, most SBOM tools support multiple SBOM formats, while as for SBOM import, there are fewer products that support multiple SBOM formats. It is desirable to determine which SBOM formats need to be supported, based on the scope of SBOM application.</p>
Components analysis method	<p>Components in software can be analyzed in three major ways: code matching, dependency detection, and string detection. Code matching is a method to detect OSS by matching a code with OSS databases. In addition to exact matching, there is a partial matching method called snippet matching. There is also a method of matching by binary patterns. Dependency detection is a method to detect direct and indirect OSS obtained with a package manager; the possibility of false detections is low. String detection is a method to detect applicable licenses by analyzing software license strings. Some SBOM tools combine multiple analysis methods for OSS analysis and some tools support only some analysis methods. Therefore, it is desirable to organize and clarify which OSS analysis method should be adopted, based on the code information that can be prepared when creating SBOM.</p>

Viewpoints	Description
Support systems	As for commercial tools, there are SBOM tools that allow users to inquire the vendor about the implementation and operation of the tool. As an option, some tools offer a plan that allows users to consult with the vendor on various aspects of OSS management, not limited to inquiries about the tools. Some companies provide support services for OSS tools as well, the users can receive assistance as needed. It is desirable to determine the level of support needed, taking into consideration the scope of the SBOM application and the knowledge level of the personnel in charge of SBOM introduction.
Coordination with other tools	There are SBOM tools that can be integrated with the development environment, build tools, software version control tools, communication tools, etc. For improving the efficiency of the entire software development life cycle, such as automation of SBOM creation, it is desirable to be able to integrate with tools already in use in the organization. It is also desirable to clarify what kind of tools need to be linked with.
Form of provision	There are two types of SBOM tools: packaged version and cloud version. With packaged SBOM tools, there is a possibility of incurring server maintenance costs in addition to tool fees, and the environment in which the tool can be deployed may be limited. With the cloud version, the initial installation cost and workloads required for SBOM sharing can be reduced compared to the packaged version. However, it is necessary to confirm in advance that there is no risk of transmitting externally highly confidential source code information of the company. It is desirable to determine which type of provision is more suitable for the organization, considering the system constraints within it.
User interface	Some SBOM tools provide only a command line interface (CLI), while others also provide a graphical user interface (GUI). GUI-compatible tools enable the intuitive creation of SBOM and visualization of the output results. It is desirable to determine what kind of user interface tools are required, taking into consideration the knowledge level of the personnel in charge of SBOM introduction.

Viewpoints	Description
Operation method	When developers execute a component analysis with SBOM tools by themselves, they can reduce their workload by selecting SBOM tools that are linked to their development environment and automatically perform analysis in the background. On the other hand, if a specialized team such as an analysis team executes an SBOM tool, it will be easier to examine the analysis results by selecting an SBOM tool that provides sufficient supplementary information such as policy functions and licenses.
Supported software language	SBOM tools support different software development languages; many tools support representative languages such as C, C++, Java, Python, Ruby, Swift, Go, etc. For some languages, however, the number of tools that support them may be limited. Based on information collected concerning the target software, it is desirable to determine which SBOM tools should be implemented for which development languages.
Japanese support	Currently, most SBOM tools are developed overseas. Therefore, in some cases, instruction manuals and README files are provided only in English, and in other cases, the tools themselves do not support Japanese. If it is difficult to use tools provided only in English, it may be better to consider prioritizing tools with Japanese support, after considering the purpose of the organization's SBOM introduction and other points of view ²⁴ . Some sales agents of commercial tools or companies that provide support for OSS tools may provide documents related to SBOM tools translated into Japanese.

Based on the purpose and scope of the SBOM introduction, it is desirable to evaluate and select SBOM tools after determining in advance what level of content is required for each point of view. For example, if the budget available for SBOM introduction is limited, it is expected to select an OSS SBOM tool that is compatible with the company's development language and capable of outputting an SBOM in the desired format. If enough budget is allocated to implement commercial tools, it is expected that multiple tools will be evaluated and selected based on

²⁴ For example, if there is a possibility of joint operation of SBOM tools with the company's overseas offices, overseas business partners, foreign suppliers, etc., it is desirable to select tools based on their functions, performance, and operation methods rather than prioritizing Japanese support.

comprehensive consideration of functionality, performance, cost, and other factors. It should be noted that the use of multiple SBOM tools may be inefficient, except in cases where each business unit or development project has a different viewpoint on the optimal tool to be sought.

In the evaluation and selection of tools, it is expected that agents who handle SBOM tools will be consulted. By experiencing the actual feeling of use and evaluating the difficulty and required period of operation learning, using a free trial²⁵ before implementing the SBOM tool, it is possible to perform a trial analysis of the source code of a project that is assumed to be a typical product or application in the company and to check whether the expected results are obtained. In addition, if it is difficult to set or select viewpoints, the organization should consult with distributors who handle multiple SBOM tools and select one by comparing and evaluating the features, advantages, and disadvantages of each tool, while obtaining information about many tools.

4.3. SBOM tools installation

[Actions for the introduction of SBOM]

- ☐ Check the requirements of the environment where the SBOM tool can be installed and set up the environment.
- ☐ Check the instruction manual and README file of the tool and then implement and configure an SBOM tool.

[Points to be aware of for SBOM introduction]

- In the case of commercial SBOM tools for which a support system is in place, the implementation and configuration of a tool can be done efficiently by contacting the sales agent or tool vendor and receiving their assistance.
- OSS SBOM tools may require the burden of trial-and-error configuration because information about tool construction and configuration may be lacking. Effective implementation and configuration of an OSS SBOM tool can be achieved by obtaining assistance from companies that provide support

²⁵ It is effective to organize the functions and use cases to be evaluated before conducting a trial, and to formulate a specific trial plan.

services related to OSS tools, if necessary.

- When using an SBOM tool for vulnerability management, it is necessary to monitor the operation of the SBOM tool and to back up the data regularly to prevent the SBOM tool from stopping due to failures or other reasons and to prevent vulnerability detection from being delayed.

The environment in which the SBOM tool can be installed differs depending on the SBOM tool. For example, the PC on which an SBOM tool runs may be required to have an internet connection, certain machine specifications, a specific OS, a specific browser installed, or a Java or Python execution environment. Also, some SBOM tools limit the installable OS solely to Linux, while a separate virtual machine environment may be required when installing on a Windows terminal. Therefore, when implementing and configuring an SBOM tool, it is necessary to first confirm the requirements for the implementation of the tool and build an environment for the implementation.

After the environment for SBOM tool installation is in place, the organization implements and configures the SBOM tool for SBOM production. Basically, the implementation and configuration should be done by checking the user's manual and README file. However, for commercial SBOM tools that have a well-developed support system, the implementation and configuration can be done efficiently with the help of a sales agent or tool vendor. Some sales agents offer services for environment construction and initial settings on behalf of their customers, so it is a good idea to consider using such services if necessary. Certain SBOM tools lack information about tool construction and configuration. In addition, since many OSS SBOM tools are developed overseas, the documents for reference are often available only in English. For this reason, it is assumed that the SBOM tool may be configured by trial and error, for example, by inputting sample codes and checking whether a desired SBOM is outputted or not. If necessary, companies that provide support services for OSS tools may be used in effectively implementing and configuring an OSS SBOM tool.

When using an SBOM tool for vulnerability management, it is necessary to monitor the operation of the SBOM tool and perform regular backups of data to prevent the SBOM tool from stopping due to failures or other reasons and to prevent a delay in vulnerability detection.

4.4. Learning about SBOM tools

[Actions for the introduction of SBOM]

- ☐ Learn how to use SBOM tools by checking the instruction manual and README file of the tool.
- ☐ Record know-how on how to use the tool and the outline of each function and share them within the organization.

[Points to be aware of for SBOM introduction]

- With commercial SBOM tools that have a support system, users can learn how to use the tools efficiently by making inquiries to their sales agents or tool vendors.
- By using tools through trial and error by creating sample SBOM, users can learn how to use their tools efficiently.

After an SBOM tool has been implemented and configured, it is desirable to learn how to use the tool. Basically, the user should learn how to use the tool by checking the instruction manual and the README file. With a commercial SBOM tool for which a support system is available, the user can efficiently learn how to use the tool by making inquiries to the sales agents or tool vendors. Compared to OSS tools, commercial tools are more sophisticated, and it may take time to learn all the functions. The user may check with the sales agent or tool vendor regarding the functions necessary to produce the SBOM that the organization desires to create and then learn how to use the tool by focusing on those functions. It is also effective to learn how to use the tool through trial and error by creating sample SBOM. This is especially effective in the case of tools for which information about how to use is lacking. Since the specific usage of the tool differs from organization to organization, it is desirable to record the know-how on the usage of the tool and the outline of each function identified through the learning process and to share them within the organization.

5. SBOM production and sharing phase

Based on the established environment and system, organizations are required to create SBOM and provide them as needed. This section discusses what SBOM-introducing organizations should do during the SBOM creation and sharing phase, as well as the points that SBOM-introducing organizations should be aware of.

5.1. Component analysis

[Actions for the introduction of SBOM]

- ☐ Scan the target software and analyze the component information using an SBOM tool.
- ☐ Examine the analysis log of the SBOM tool and check whether the analysis has been correctly executed without any false positives or false negatives caused by errors or lack of information.
- ☐ Check the component analysis results to see if there are any false positives and false negatives.

[Points to be aware of for SBOM introduction]

- A Component analysis function of SBOM tools can be used to analyze components and create SBOM more efficiently than the manual method. The effect of using an SBOM tool is greater when the number of components is larger.
- In some cases, it is effective to use the configuration information of a package manager. In some cases, the package manager may also be used to identify granular components that cannot be identified by the component analysis function of SBOM tool.
- False positives and false negatives of components may occur. For example, components such as symbolic links and runtime libraries, deep hierarchical components, and components used only in specific fields may not be detected. Even if components are identified, their version information may be wrong.
- The output results differ, depending on the component analysis method in the

SBOM tool. In the case of analysis based on dependencies, the possibility of false detection is extremely low, but in the case of other analysis methods, there is a possibility of false positives and false negatives. In the case of analysis based on binary files, there is an advantage that only binary files can be used for analysis even when source codes are not available. There is a possibility, however, that the accuracy of analysis will decrease when only binary files are used.

- Analysis results may differ, depending on the environment (execution environment, development environment, etc.) in which components are analyzed.
- Since OSS that does not exist in the SBOM tool database cannot be detected, additional measures may be needed, such as manually adding information about the component from the SBOM tool console.
- Component relationships in an SBOM created with an SBOM tool may differ from the actual software configuration and need to be analyzed with appropriate settings.
- It takes a particularly large number of workloads to check for false positives and false negatives related to sub-tier components and third-party components. Since it is difficult to guarantee false negatives, the check must be based on the trade-off between the degree of accuracy and the workloads required to deal with the problem.
- By considering the analysis method of the SBOM tool, false positives and false negatives can be efficiently checked.

The PoC confirmed that the component analysis function of a SBOM tool can analyze components and create SBOM more efficiently than the manual method. For example, in the PoC for dental CTs in the medical device industry, it was confirmed that manual SBOM creation required more than 30 workloads, while SBOM creation using an SBOM tool required only 0.15 workloads, leading to a reduction of 99% or more. Therefore, it is realistic to analyze components and create and manage SBOM using SBOM tools, and this section is also written assuming that SBOM tools will be used²⁶. In some cases, a package manager may

²⁶ Another method of creating SBOM is to automatically generate them when building the software. For example, the following examples of SBOM generation using build environments

be able to identify fine-grained components that cannot be identified by SBOM tools, and SBOM may be created effectively by utilizing the configuration information of the package manager. In addition, SBOM can be created efficiently by receiving SBOM from software suppliers when possible.

To produce an SBOM, the organization first scans the target software with an SBOM tool and analyzes the component information. The scanning method differs depending on the SBOM tool. In some cases, analysis is performed by specifying the target software from the GUI, while in other cases the analysis is performed via the CLI. For the method of analysis, organizations should check the instruction manual or README file of the implemented SBOM tool. By analyzing the SBOM tool, organizations can identify the names of components, supplier names, versions, and dependencies among components that are included in the target software. However, it should be noted that there may be cases of false positives and false negatives of components. In fact, the following points were found in the PoC:

- Components whose entities, such as symbolic links and runtime libraries, were not included in the SBOM tool scan, were not detected.
- Compared to the detection results for the top-level components, the false negative rate was high for the lower-level components. However, in some cases, only lower-level components were detected without top-level components being detected, indicating that the detection rate does not necessarily vary depending on the hierarchy of components.
- Components related to controls used only in specific areas were not detected.
- Several components were detected with incorrect version information.
- The output results differed depending on the component analysis method of the SBOM tool. The results of the binary scan using only binary files showed that only about 10% of the components were detected, compared to the number of components detected in the normal scan.
- The analysis results differed depending on the environment in which the components were analyzed. When a scan was performed in the development environment, uninstalled packages that were not actually used in the product

and tools are available:

- Yocto : <https://docs.yoctoproject.org/dev/dev-manual/sbom.html>
- Android Open Source Project (AOSP) :
<https://source.android.com/docs/setup/create/create-sbom?hl=ja>
- Zephyr : <https://docs.zephyrproject.org/latest/develop/west/zephyr-cmds.html>

were also detected.

- No OSS that did not exist in the SBOM tool database was detected, necessitating adjustment of the analysis results, such as manually adding component information from the SBOM tool console.
- Depending on the repository and settings of the SBOM tool, the configuration information of the components in the SBOM was different. There were cases in which the relationship of components in the SBOM created with the SBOM tool was different from the actual software configuration.
- In some cases, the components detected by the SBOM tool did not match the components extracted by the package manager.

Therefore, it is important to check the output results for false positives and false negatives, instead of using the output results of the component analysis function of a SBOM tool as they are. The viewpoints and methods of checking the results for false positives and false negatives are shown in Figure 5-1. In some cases, it is practically difficult to check all the components comprehensively, because the confirmation of false positives and false negatives is basically a manual process. In the PoC, some components require 0.50 hours/component to check for false positives and false negatives, which means that many workloads are required to check for false positives and false negatives in the case of software with many components. Checking for false positives and false negatives related to sub-tier components and third-party components requires many workloads. Since it is difficult to guarantee the absence of false negatives, checks should be based on the trade-off between the degree of accuracy and the support workloads.

Points to check		How to check
Is the tool operating properly?		(i) Check the execution log of the tool for errors.
Are the component analysis results correct?	Are components being mis-detected?	(ii) Check whether the component being output is included in the target software. If the match type is not an exact match, check whether the component may have been modified. (iii) (If the decision cannot be made in (ii)) Based on the component name outputted, compare the source code obtained from external sources such as GitHub with the source code of the target software.
	Are the components included (no omissions detected)?	(iv) Check whether the components included in the target software are included in the output results.
	Are modified components detected?	(v) Based on the results of snippet analysis, compare the source code of the target software with the original source code.
	Are there any undetected components?	(vi) Extract the copyright of the target software and check if the detected components are included in the output results

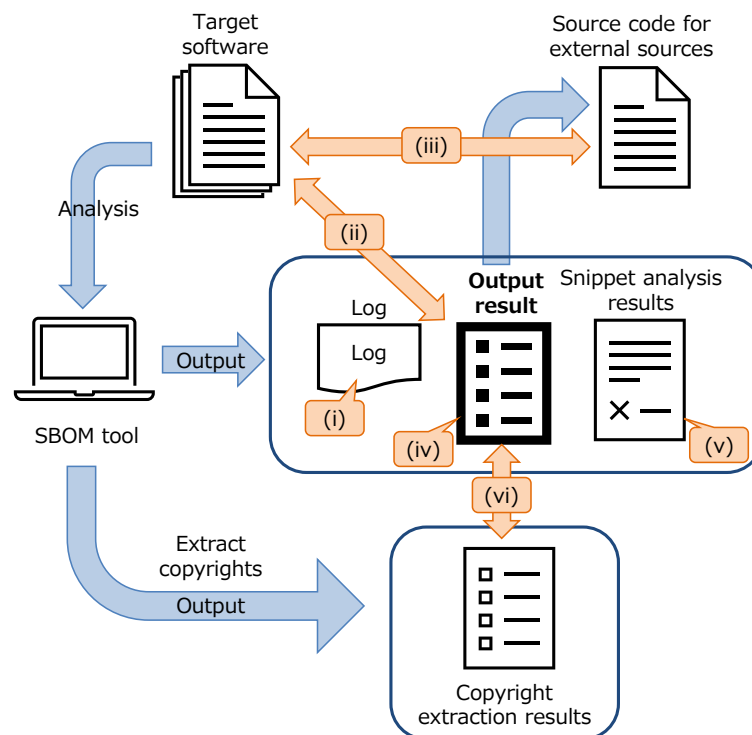


Figure 5-1 Perspectives and methods of checking component analysis results

When checking for false negatives, it is important to consider the analysis method of the SBOM tool. There are three major methods of component analysis in SBOM tools: code matching, dependency detection, and string detection. Dependency relationship detection is a method to detect direct and indirect OSS obtained by a package manager; the possibility of false detections is low. On the other hand, in

the case of analysis by code matching or string detection, there is a possibility of false positives and false negatives. In addition, in the case of scans based on binary files, it was found that many false negatives have occurred. Since the degree of occurrence of false positives and false negatives varies depending on the component analysis method, it is desirable to check for false positives and false negatives based on the analysis method of the SBOM tool in use. For example, in the case of analysis based on binary files, there is an advantage that only binary files can be analyzed even when source code is not available. On the other hand, it is expected that false positives and false negatives are checked for, while considering, among other things, the possibility that many false positives and false negatives may occur when only binary files are used. There is also a possibility that the analysis is not being performed properly due to insufficient parameter settings of the SBOM tool, failure of package manager execution, or other reasons, resulting in false positives and false negatives. Even if the tool seems to be terminated normally on the surface, it may be terminated by skipping a part of the internal analysis process due to an error. Therefore, it is necessary to check the execution log of the tool to see if such an error has occurred.

If, because of the confirmation of false positives and false negatives, it is found that unknown information is contained, it is desirable to understand such information as “known unknowns”. Known unknowns are facts that are unknowns but considered as knowns, which are also referred to in the “Minimum Elements” of the SBOM, as shown in Table 2-3. When sharing a created SBOM with users and suppliers of the target software, the transparency of the information can be enhanced by sharing the “known unknowns” as well.

5.2. SBOM production

[Actions for the introduction of SBOM]

- ☐ Determine the requirements for the SBOM to be produced, such as items, format, and output file format.
- ☐ Produce an SBOM that satisfies the requirements, by using the SBOM tool.

[Points to be aware of for SBOM introduction]

- Considering the purpose of creating and sharing an SBOM, full accurate

information should be included in the SBOM.

- When a component is used that is provided by a third party, such as an OSS community, it may be able to receive an SBOM of the component. However, if the component is used after being modified within the organization, it will not be able to use the provided SBOM as it is.
- By setting the names in the SBOM from the viewpoint of SBOM users, it is possible to eliminate rework after the SBOM is shared.

Produce an SBOM based on the analyzed component information. When creating an SBOM, it is necessary to determine in advance the requirements regarding the SBOM, such as the items to be included in the SBOM, the format, and the output file format. For these requirements, regulations/requirements may specify the format and items of the SBOM. In the SBOM format, no information (NOASSERTION) is allowed, but considering the purpose of creating and sharing the SBOM, it is desirable that the correct information is fully entered in the SBOM. If a component provided by a third party such as a third party or OSS community is used, organizations may be able to receive the SBOM for the component. By receiving SBOM from a third party, it is possible to create SBOM efficiently, and organizations may also use them to examine the SBOM created by the company. It should be noted that there are contractual and licensing issues regarding whether to request communities or individuals to provide SBOM. To identify any rewriting or unauthorized tampering of components in the supply chain, it is effective to check the consistency of the SBOM provided by a third party and those created by your own organization. In addition, users and suppliers of software that may share an SBOM may specify the SBOM. It is necessary then to determine the requirements for the SBOM in consideration of their situation. Since the specific SBOM creation method differs depending on the tool, please refer to the user's manual or README file of the SBOM tool implemented.

SBOM should not only be created but also be managed continuously, and the date and time of creation of an SBOM should be clearly recorded. To enhance the transparency of the software supply chain, it is desirable to share a created SBOM as necessary with the users and suppliers of the target software. In sharing the SBOM, it is required to confirm that the necessary information is included.

The created SBOM may include not only component information but also information configured on the SBOM tool, such as project name. It is desirable to consider whether this information is easy for SBOM users to utilize. When

components are managed with the SBOM tool from the development stage, project names and version information used there are reflected in SBOM. There is then a possibility that information that was previously used only within the company will be shared with SBOM users. By setting names in the SBOM that can be understood by SBOM users, it is possible to eliminate rework after sharing the SBOM.

5.3. SBOM sharing

[Actions for the introduction of SBOM]

- ☐ Share an SBOM with the users and/or suppliers of the target software as necessary after determining the method of sharing the SBOM.
- ☐ Consider using electronic signature technology or other technologies to prevent falsification of the sharing of SBOM data.

[Points to be aware of for SBOM introduction]

- Different SBOM sharing methods may be adopted, depending on the SBOM tool used by the supplier.
- Various SBOM sharing methods will be available to different users. When sharing an SBOM with users, it is necessary to examine the advantages and disadvantages of each SBOM sharing method.

From the viewpoint of increasing the transparency of the software supply chain, it is desirable as necessary to share a created SBOM with users and suppliers of software. When sharing an SBOM is required by regulations or requirements, it is necessary to share the SBOM with appropriate parties in an appropriate manner in accordance with the contents specified in the regulations or requirements. When considering an SBOM sharing method, it should be noted that the contents of many SBOM change dynamically after their creation due to the version-up of components. As described in Section 2.5, it is not mandatory to disclose SBOM. SBOM creators and suppliers are encouraged to decide how to share SBOM at their own discretion.

When sharing an SBOM with suppliers, the sharing method varies, depending on the SBOM tool used by the supplier. In general, if an organization and the recipient use the same SBOM tool, it is relatively easy to share the SBOM with the recipient.

Especially in the case of commercial SBOM tools, the SBOM can be shared between the organization and its users or suppliers by using the same SBOM tool in the cloud. On the other hand, if the organization, users, and suppliers use different SBOM tools, there may be restrictions on the SBOM formats, depending on the tools. It is desirable to discuss SBOM sharing methods and contents of a shared SBOM with suppliers, in advance. Currently, there are only a limited number of tools that can import SBOM generated with other tools and use them for vulnerability management. Therefore, care should be taken when discussing with users and suppliers.

Various methods may be available for SBOM sharing with users. For example, an SBOM sharing method may be integrated into the product so that the SBOM can be checked from within the product; the SBOM sharing method may be published in a repository accessible to users; or a common SBOM tool may be used for sharing SBOM data. When sharing SBOM with users, it is desirable to select an SBOM sharing method, considering the characteristics and frequency of updates of SBOM target software, SBOM usage status among users, and so on. In addition, to ensure the reliability of SBOM data itself when sharing an SBOM, it is necessary to consider the use of digital signature technology, distributed ledger technology, or other technologies to prevent tampering.

6. SBOM use and management phase

To enjoy the benefits of SBOM, it is required to operate and manage SBOM that have created. This section shows the items that the SBOM-implementing organization should implement, as well as the points that SBOM-implementing organizations should note, in the SBOM operation and management phase.

6.1. Vulnerability management, license management, etc.

[Actions for the introduction of SBOM]

- ☐ Based on the output of the SBOM tool, assess the severity, evaluate the impact, fix the vulnerabilities, check the residual risk, and provide information to the relevant organizations.
- ☐ Based on the output of the SBOM tool, check whether there is any violation of the OSS license.

[Points to be aware of for SBOM introduction]

- The vulnerability information and license information outputted by the SBOM tool may be incorrect, so it is necessary to check the output results.
- If the EOL of a component cannot be identified by the SBOM tool, it is necessary to investigate it separately.

In this phase, vulnerability management, license management, etc. are performed based on the created SBOM. As mentioned above, Since SBOM is a method of software management, the aim should not be to create an SBOM itself, but to use SBOM to achieve appropriate software management. Therefore, vulnerability management and license management need to be implemented on SBOM data provided by third parties. In vulnerability management, it is necessary to check, based on the outputs of the vulnerability management function of the SBOM tool, whether the components included in the software are vulnerable or not. If a vulnerability is found, countermeasures must be taken against it. As a specific vulnerability response, it is desirable to locate the vulnerability, analyze the scope of impact, estimate and evaluate the risk, confirm the acceptability of the risk, and

prioritize the vulnerability response. Then, after identifying the related security issues, it is desirable to evaluate the severity of the vulnerability and decide on urgency. When a vulnerability is identified in the proprietary software of the company, the related software users should be notified appropriately. When a vulnerability is identified in third-party components such as OSS and general-purpose software, the vulnerability should be notified to the suppliers of those components. It should be noted that in the analysis of the impacted area of vulnerability, it is necessary to identify and analyze not only the source code but also development documents such as requirement definitions, specifications, and test specifications that need to be updated. As an example of countermeasures for this point, the PoC conducted in FY2021 confirmed that it was possible to reduce the workloads required for identifying the affected scope of vulnerabilities, by linking the SBOM tool with an existing configuration management tool.

When managing an SBOM manually, it is necessary to manually identify each vulnerability, assess each vulnerability individually, and consider how to respond to each vulnerability. Since vulnerability information is updated daily, manual operation and management of an SBOM is impractical. Therefore, as shown in Figure 6-1, SBOM tools are expected to be used for vulnerability management as well. It should be noted that there is a large difference between commercial and OSS SBOM tools in the range of vulnerability matching. Some OSS tools do not have a vulnerability matching function, while some commercial tools have enhanced vulnerability information databases such as NVD and JVN, as well as their own vulnerability information database, to expand the scope of vulnerability matching. Some commercial SBOM tools automatically match analyzed components with vulnerability information and information about the severity, risk, and remedies of the vulnerabilities, thus making it possible to quickly find vulnerabilities, assess their severity, and determine remedies. However, even if vulnerability information is identified, if specific remedies are not provided, it is necessary to consider remedies separately based on the details of each vulnerability.

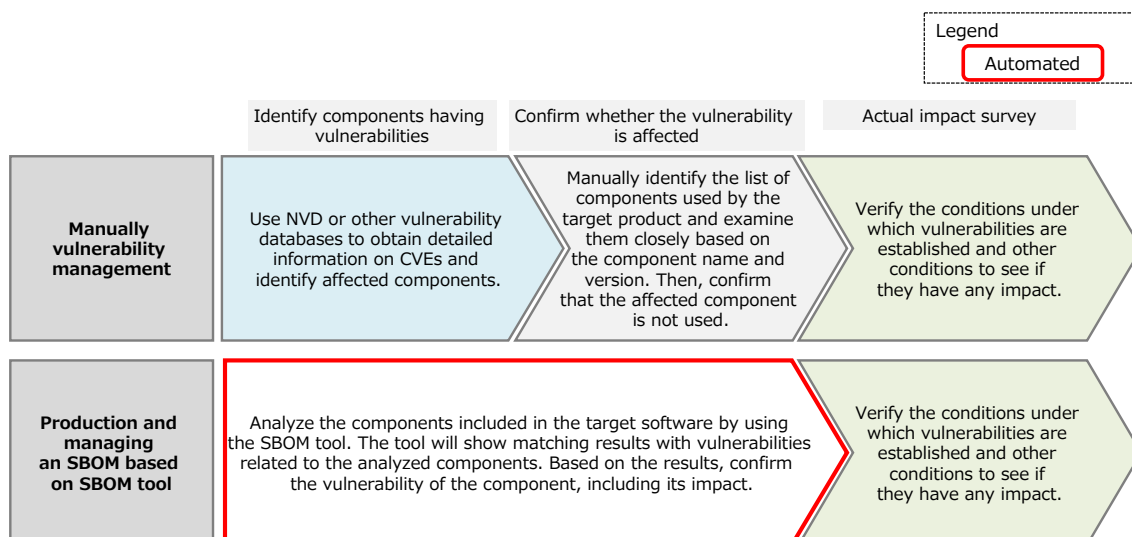


Figure 6-1 Comparison of vulnerability management procedures followed manually or with an SBOM tool

One of the points to note when managing vulnerabilities based on an SBOM tool is that the vulnerability information outputted by an SBOM tool may contain errors. In some cases, the OSS SBOM tools used in the PoC outputted incorrect information about the severity of vulnerabilities, and it was necessary to manually investigate the vulnerability information. There is a possibility of false positives and false negatives in the analysis of components; there is also a possibility of errors in the output results of vulnerability information. It is then necessary to check the output results. Some SBOM tools perform vulnerability matching based on not only vulnerability information in public vulnerability information databases such as NVD but also vulnerability information based on tool vendors' own surveys, which may enable vulnerability management based on a wide range of vulnerability information.

In response to these perspectives and issues, Section 7 summarizes specific procedures and methods for vulnerability management using SBOM, divided into process phases.

Based on the outputs of the SBOM tool, it is necessary to check the license compliance status of the components included in the software. If it is determined that it is impossible or difficult to comply with the license conditions regarding the assumed usage of the component in question, it is necessary to take measures such as changing the component itself or the usage method. As in the case of vulnerability management, it is more practical to use SBOM tools instead of manual

management.

SBOM tools can efficiently identify vulnerabilities and license information of components included in the target software. It is generally difficult to identify the EOL of components using tools, and it is necessary to identify them manually. Since there are some components that have no information about EOL, it is desirable to avoid using such components as much as possible.

6.2. SBOM information management

[Actions for the introduction of SBOM]

- ☐ Keep the created SBOM for a certain period, including the change history, so that it can be referred to in case of inquiries from outside the company.
- ☐ Manage the information contained in the SBOM and the SBOM itself appropriately.

[Points to be aware of for SBOM introduction]

- Information about new vulnerabilities can be immediately obtained by using an SBOM tool that automatically updates and notifies vulnerability information. If automatic management using a tool is not possible, it is necessary to cover the situation in terms of operation by appointing a separate person in charge, but this requires more workload.
- SBOM can be most effectively managed by the department corresponding to PSIRT in the organization, or by the quality control department if there is no department corresponding to PSIRT.

The created SBOM shall be retained for a certain period, including a change history so that they can be referred to in case of inquiries from outside the company. The SBOM should be retained for a minimum period while the product is generally distributed in the market. Even after the end of sales, it is necessary to maintain SBOM for reference in advance because they may be referred to as necessary during the warranty period, support provision period, replacement components provision period, and so on. In addition, if there is an individual specification in the license conditions of the component used, such as three years after the end of product provision, the period should also be taken into consideration. It is also

assumed that the SBOM modification history will be stored in the asset management system so that the SBOM information can be associated with the shipped products.

Given that the contents of software covered by an SBOM change dynamically and that vulnerability information is updated daily, the information contained in the SBOM needs to be updated and managed periodically. By using an SBOM tool that automatically updates and notifies vulnerability information, information about new vulnerabilities can be immediately obtained. If automatic management using a tool is not possible, operations must be covered, for example, by separately appointing personnel to be in charge. In such a case, it should be noted that it requires more workload than management with SBOM tools.

Regarding the SBOM management system, it is desirable from the viewpoint of vulnerability management that the PSIRT or a similar department in the organization take the lead in SBOM management. In addition, by utilizing created SBOM, PSIRTs can reduce the workloads required for narrowing down the OSS used in users' environments, thus enabling more efficient vulnerability countermeasures and monitoring. Even if there is no department equivalent to PSIRT, vulnerability management should be conducted under a certain policy, for example, SBOM being managed by the quality control department. If there is a team in charge of quality control across the company, it would be possible to operate under a certain policy by defining and managing SBOM as a deliverable and addressing vulnerabilities by utilizing SBOM as part of quality control. If there is no quality control department, it is expected that a specific product development team will first implement an SBOM tool and then accumulate know-how concerning the creation, operation, and management of the SBOM. After that, it is desirable to improve the level of SBOM introduction in the company by horizontally deploying the obtained know-how to other development teams to promote SBOM introduction into each team.

7. Specification of Vulnerability Management Process

7.1. Purpose

One of the security benefits of utilizing SBOM is the reduction of the potential for exploitation through vulnerability management, which involves identifying and addressing vulnerabilities. Therefore, among the overall processes of creating, sharing, operating, and managing SBOM, the phases related to vulnerability management are particularly important. This chapter focuses on the vulnerability management process using SBOM, summarizing specific procedures and considerations to provide reference information that enhances the effectiveness of SBOM.

7.2. Challenges and issues in vulnerability management

In promoting the efficiency and widespread adoption of vulnerability management utilizing SBOM, various stakeholders—such as equipment manufacturers, component suppliers, and user organizations—are involved. There are several challenges related to technology, standards, and procedures within the vulnerability management process. The figure below illustrates the main challenges across the vulnerability management process as the horizontal axis, providing an overview of the situation.

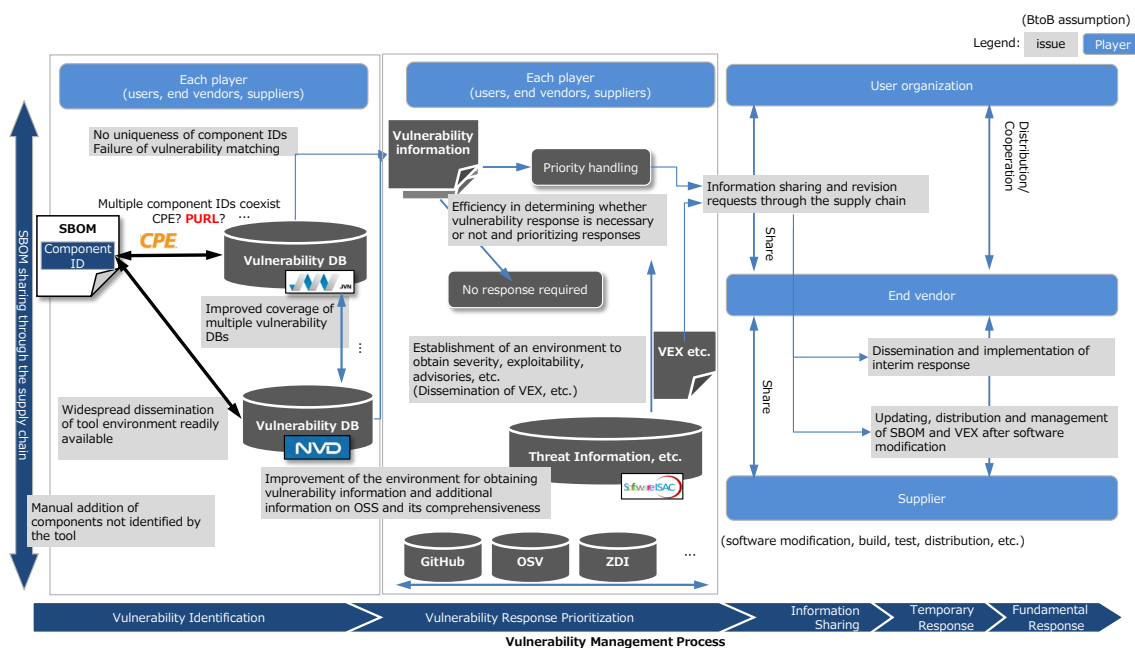


Figure 7-1 Challenges in vulnerability management utilizing SBOM (Overview)

The vulnerability management process utilizing SBOM can be implemented in phases as shown on the horizontal axis of the figure: vulnerability identification, vulnerability response prioritization, information sharing, and vulnerability response (including temporary and fundamental response). Since there are challenges associated with vulnerability management using SBOM, this section will outline the issues present in these phases and present methods and procedures to address them in the following section.

First, in the vulnerability identification phase, the lack of uniqueness due to various standards and vendor-specific formats of component IDs included in the SBOM poses a challenge for vulnerability matching. Additionally, multiple vulnerability databases exist, and expanding the scope of these databases is necessary to enhance the comprehensiveness of vulnerability information. In the vulnerability response prioritization phase, determining whether a response is necessary and setting priorities are critical for improving efficiency. The acquisition of required information from external sources and the dissemination of VEX (Vulnerability Exploitability Exchange) information also present challenges. In the information sharing phase, identifying the scope, methods, and environment for information sharing becomes a challenge. During the vulnerability response phase, considerations for temporary measures that do not involve fixing vulnerabilities, as well as updating and sharing SBOM and VEX information based on the results of vulnerability fixes, are key issues.

The following chapters will outline methods and procedures to address these challenges, demonstrating ways to achieve effective vulnerability management utilizing SBOM.

7.3. Overview of the entire process

Considering the challenges outlined in the previous section, the overall view of methods and procedures essential for implementing vulnerability management using SBOM can be summarized as follows:

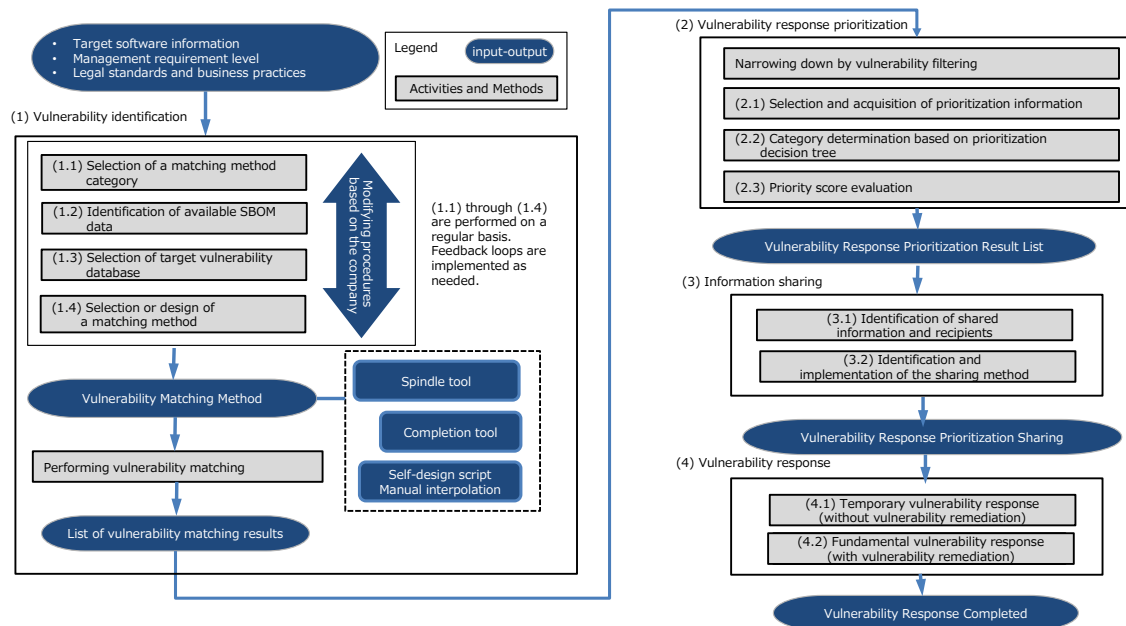


Figure 7-2 Key Steps and Procedures in the vulnerability management process utilizing SBOM (Overview)²⁷

The vulnerability management process utilizing SBOM consists of the following four steps, as illustrated in the figure. This section provides reference examples of important methods and procedures for each step.

(1) Vulnerability Identification Phase

Use SBOM to identify vulnerabilities present in the software based on the latest vulnerability information.

(2) Vulnerability Response Prioritization Phase

Assess the identified vulnerabilities to determine the necessity and priority of responses based on the potential for exploitation and cost-effectiveness.

(3) Information Sharing Phase

Share information regarding vulnerabilities and response methods among stakeholders.

(4) Vulnerability Response Phase

Implement rapid temporary measures for prioritized vulnerabilities that do not involve fixing them, as well as permanent solutions that include vulnerability fixes. Update and share SBOM and VEX information based on the results.

²⁷ The organization of the vulnerability management process is based on references such as the following documents:

NTIA, "Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)", CISA, "Stakeholder-Specific Vulnerability Categorization (SSVC)", CISA, "SBOM Sharing Roles and Considerations"

7.4. Procedures and methods for each phase

In this chapter, specific procedures, and methods for each phase of vulnerability management utilizing SBOM will be presented, as outlined in the previous chapter. This aims to provide examples that organizations can reference to implement vulnerability management according to their policies and environments.

7.4.1. Vulnerability Identification Phase

Vulnerability identification primarily involves determining the methods for identifying vulnerabilities within the organization using one of the following four approaches. Actual vulnerability identification is then performed using the selected method. Since the necessary items and order may differ based on the organization, it is expected that each organization will selectively implement these methods.

Selection of matching method category

(1) Selection of a matching method category

Vulnerability matching methods can be categorized into: Use of (i) existing SBOM tools, (ii) API utilization scripts, and (iii) Web UI. Organizations should choose a method based on their technical capabilities, budget, and available resources.

(2) Identification of available SBOM data

Determine how to obtain the SBOM that will be utilized.

(3) Selection of target vulnerability database

Choose the vulnerability databases that will be used for vulnerability identification and prioritization of responses.

(4) Selection or design of a matching method

Based on the results from steps (1) to (3), decide on the organization's specific method for vulnerability identification.

(1) Selection of a matching method category

As shown in the figure below, vulnerability matching methods can be categorized based on the configuration of the client-side and the vulnerability database side into three types: (i) existing SBOM tools, (ii) API utilization scripts, and (iii) Web UI.

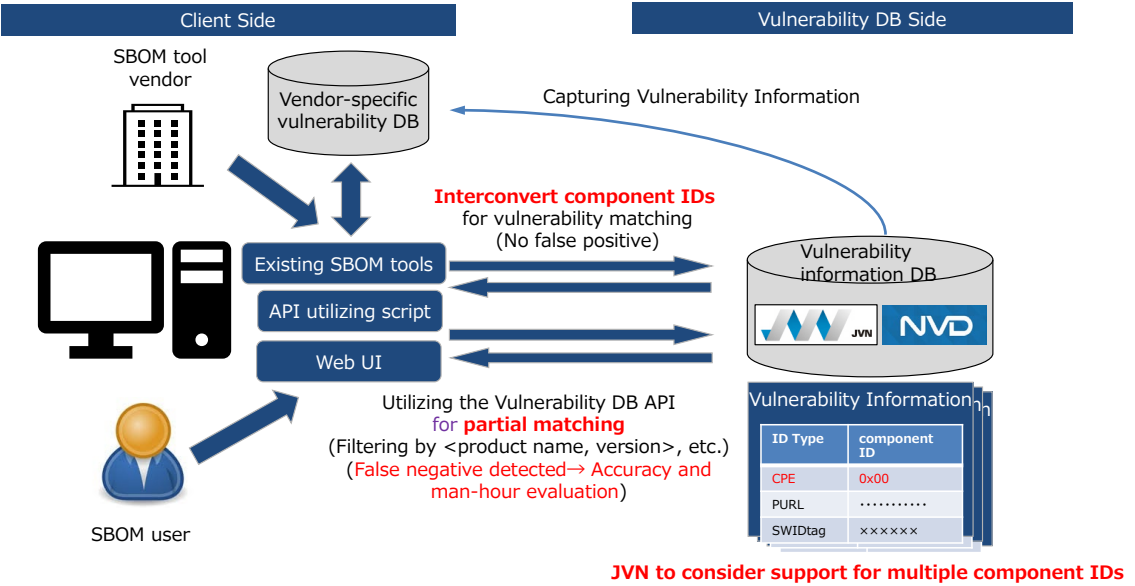


Figure 7-3 Overview and options for vulnerability matching categories

The vulnerability matching categories can be organized as follows, including use cases, main users, and their merit and demerit. Organizations and users are expected to choose methods based on their specific circumstances, using this information as a reference.

Table 7-1 Organization of use cases, main users, advantages, and disadvantages of vulnerability matching categories

Matching category	Use case/necessity	Main user	Benefits/ Drawback
Using API²⁸	By using an API for vulnerability searches on vulnerability databases that cannot be searched with existing tools, the comprehensiveness of vulnerability detection can be enhanced.	Software vendors (manufacturers and suppliers) that require a high level of demand for their software (with low vulnerability risk) are expected to be the main users. Additionally, high-demand user companies, such as critical infrastructure operators, may also require autonomous and comprehensive vulnerability searches using APIs	<p>(Merit) By utilizing the API, processes can be flexibly customized such as part ID conversion and expanding the scope of the vulnerability database, enabling continuous automated monitoring. This allows for the development of search methods that prevent missed detections and false positives, as well as linking alerts post-search.</p> <p>(Demerit) The use of the API requires technical skills in coding and resources in terms of workload and other factors.</p>

²⁸ This refers to APIs provided by vulnerability databases. For example, a public example is the API offered by MyJVN.

Matching category	Use case/necessity	Main user	Benefits/ Drawback
Using existing tool	With limited personnel and technical resources, it will be focused on addressing the minimum necessary vulnerabilities from vulnerability databases that can be searched using existing tools.	For paid tools, large enterprises with a sufficient budget are the target users. For free tools, the target users include small and medium-sized organizations and vendors, which often have limited personnel and technical resources for API coding	<p>(Merit) Vulnerabilities can be identified with limited personnel, without the need for API coding.</p> <p>(Demerit) The scope is limited to vulnerabilities present in the databases provided by existing tools, resulting in a limited comprehensiveness of identified vulnerabilities.</p>

Matching category	Use case/necessity	Main user	Benefits/ Drawback
Using Web UI	Before coding with the API, methods are considered for vulnerability search and assesses the vulnerability status preliminary to keep the workload minimal.	API users who require a high level of security, as well as users of existing tools, will utilize it for preliminary vulnerability searches before regular operations	<p>(Merit) Before regular operations through API coding or existing tools, the user can conduct trial evaluations of vulnerability searches and assess the current state of vulnerabilities.</p> <p>(Demerit) The use of the Web UI requires manual operations, making it less efficient compared to utilizing APIs or existing tools that can be automated for regular operations.</p>

(2) Identification of available SBOM data

Available SBOM data can be identified based on the following considerations:

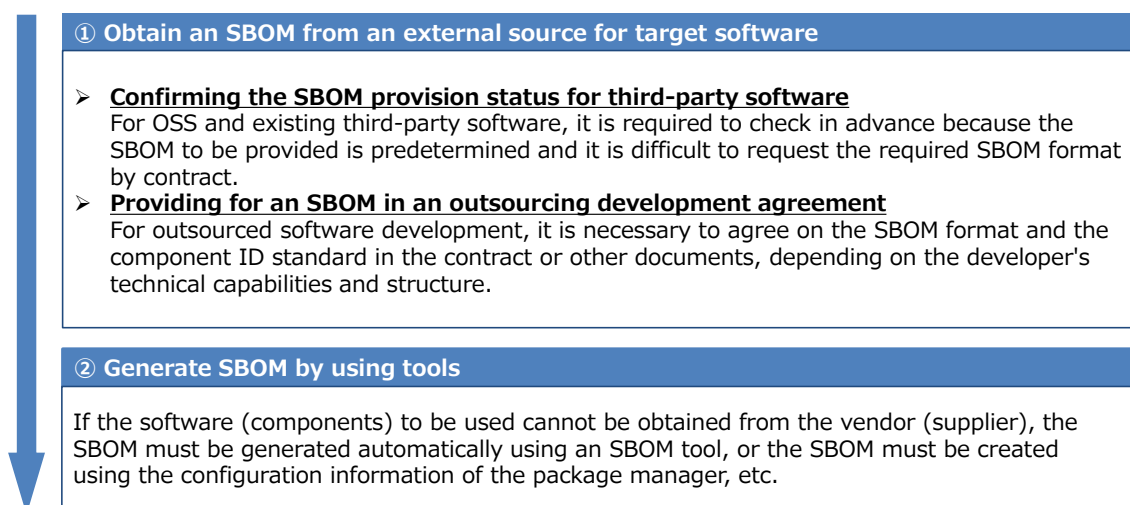


Figure 7-4 Identification of available SBOM data

If a supplier ensures adequate vulnerability management for third-party software, the priority for obtaining SBOM for vulnerability management by the procuring organization may be lower. However, since there is no guarantee that the supplier will continue to operate as a business, obtaining SBOM is important for conducting vulnerability management independently. Additionally, acquiring SBOM is necessary for other uses, such as license management and configuration management, beyond just vulnerability management.

The following formats are candidates for representing SBOM as typical examples:

SBOM format	Development entity	Characteristics
SPDX	Linux Foundation	Standardized primarily for intellectual property and license management. Capable of managing packages, containers, snippets, and other targets.
CycloneDX	OWASP	Developed primarily for security management. Can encompass VEX.
SWID	ISO/IEC, NIST	A standard for software management that includes a software ID system.

The following standards for component IDs are considered representative candidates:

Component ID	Development entity	Characteristics
PURL	OSS community (gitter)	A distributed allocation method where IDs are determined based on the repository, primarily centered around package managers like OSS.
CPE	NIST	It is specified as an element of the security information sharing standard SCAP, with CPE being assigned mainly when vulnerabilities are reported.
SWID	ISO/IEC, NIST	A superset of CPE. While the NVD has declared a transition from CPE to SWID, adoption has not progressed significantly currently.

It is recommended to verify at what stage the SBOM being utilized reflects the target software. SBOM can differ in comprehensiveness and other aspects based on the timing of their creation, such as during source code development,

compilation, or runtime.²⁹ When obtaining SBOM data, it is advisable to define which creation timing of the SBOM data is necessary based on the nature of the target system or software for vulnerability response. Additionally, it is recommended to confirm when the actual SBOM data was created and to adjust the necessary tasks accordingly to ensure accurate vulnerability response. Furthermore, it is important to verify that the SBOM being utilized is applicable to the version of the target software. Using SBOM for different versions may result in incorrect reflection of the software components included in the vulnerability matching target version, potentially leading to false positives, or missed detections in the vulnerability matching process.

(3) Selection of target vulnerability database

The selection of vulnerability databases is expected to involve a comparison from the perspectives of risk reduction and cost efficiency, considering each organization's priority policies. When choosing a vulnerability database, considerations should include the expansion of vulnerability information coverage, the acceleration of vulnerability response, and cost-effectiveness. The priority of these factors may vary between organizations, so it is anticipated that each organization and product will select the appropriate scope of vulnerability databases. For example, small and medium-sized enterprises with tight budget constraints may prioritize cost and ease of use when selecting a DB, while companies with a high demand for risk reduction may prioritize databases that offer expanded coverage and quicker response capabilities.

²⁹ CISA, "Types of Software Bill of Materials (SBOM) Documents"

Individual companies consider priority policies

Comparative priorities			Top rated vulnerability DB			Standard example
Risk reduction	Expanding coverage of vulnerability information	Number of vulnerabilities	Private DB1	Public DB1	Public DB2	Over 70% of cases covered
		Non-CVE Vulnerabilities	Private DB2	Private DB3	Private DB4	Targets other than CVE
		Emphasis on Japanese products	Public DB2	-	-	Largest set made in Japan
		Emphasis on OSS	Private DB4	Private DB2	Private DB5	Prioritization and clarification
Risk reduction	Expediting and streamlining vulnerability responses (Prioritization)	Incident status	Public DB1	Private DB1	Private DB8	Dedicated field available
		Exploit status	Private DB6	Private DB7	Private DB1	Dedicated field available
		CVSS status	Public DB1	Public DB2	Private DB1	Top 3 required
		Advisory status	Public DB1	Public DB2	Private DB1	Dedicated fields, top 3
Cost reduction	Automation	Component ID standard	Public DB1	Public DB2	Private DB1	Standard designation
		API and tool provision	Public DB1	Public DB2	Private DB2, Private DB4	API available
		Easy to create scripts	Public DB1	Public DB2	-	Standard ID ^ API available
		Providing information for free	Public DB1	Public DB2	Other 7 cases	2 free of charge
		Providing Japanese language information	Public DB2			

Legend: Select targets as a union set in the frame

Figure 7-5 Comparative image of considerations for selecting target vulnerability databases

(4) Selection or design of a matching method

Based on the categories of matching methods, the format of the input SBOM, and the results of the selected vulnerability databases and their constraints, organizations should determine which methods are available for use. The table below provides reference judgments based on specifications and example validations at the time of the empirical study.

			Vulnerability DB										
Method Category	SBOM format	Component ID Standard	Public DB1	Public DB2	Public DB3	Private DB1	Private DB2	Private DB3	Private DB4	Private DB5	Private DB6	Private DB7	Private DB8
API	SPDX	CPE	x	o	o	o	o	o	o	x	x	x	o
		PURL	△	△	o	o	o	o	o	x	x	x	o
		Vender Specific ID	△	△	o	o	o	o	o	x	x	x	o
	Cyclone DX	CPE	x	o	o	o	o	o	o	x	x	x	o
		PURL	△	△	o	o	o	o	o	x	x	x	o
		Vender Specific ID	△	△	o	o	o	o	o	x	x	x	o
	SWID	SWID	x	x	x	x	x	x	x	x	x	x	x
		CPE	x	o	o	o	o	o	o	x	x	x	o
		PURL	△	o	o	o	o	o	o	x	x	x	o
		Vender Specific ID	△	o	o	o	o	o	o	x	x	x	o
Web UI	SPDX	CPE	o	o	o	o	o	o	o	o	o	x	o
		PURL	o	o	o	o	o	o	o	o	o	x	o
		Vender Specific ID	o	o	o	o	o	o	o	o	o	x	o
	Cyclone DX	CPE	o	o	o	o	o	o	o	o	o	x	o
		PURL	o	o	o	o	o	o	o	o	o	x	o
		Vender Specific ID	o	o	o	o	o	o	o	o	o	x	o
	SWID	SWID	x	x	x	x	x	x	x	x	x	x	x
		CPE	o	o	o	o	o	o	o	o	o	x	o
		PURL	o	o	o	o	o	o	o	o	o	x	o
		Vender Specific ID	o	o	o	o	o	o	o	o	o	x	o
Existing tools	SPDX (json)	CPE	x	x	x	x	x	x	x	x	x	x	x
		PURL	x	x	x	x	o Tool3	o Tool3	x	x	x	x	x
		Others	x	x	△ Tool2	o Tool2	x	x	x	x	x	x	x
	Cyclone DX (json)	CPE	x	x	x	x	x	x	x	x	x	x	x
		PURL	x	x	x	x	o Tool3	o Tool3	x	x	x	x	x
		Others	x	x	△ Tool2	o Tool2	x	x	x	x	x	x	x
	SWID	CPE	x	o Tool1	x	x	x	x	x	x	x	x	x
		PURL	x	x	x	x	x	x	x	x	x	x	x
		Others	x	x	x	x	x	x	x	x	x	x	x
		Others	x	x	x	x	x	x	x	x	x	x	x

Method Category · Input SBOM Data

Figure 7-6 Reference list for selecting vulnerability matching methods (Image)

In vulnerability matching, various vulnerability databases are involved. However, due to the coexistence of multiple component ID standards, challenges related to ID uniqueness for matching are expected to persist. Therefore, using APIs or Web UIs of vulnerability databases to identify vulnerability information through partial matching based on component names, vendor names, and other criteria is considered a practical approach.

Vulnerability matching may face challenges such as the lack of comprehensiveness in component information within the SBOM and the absence of uniqueness in component IDs, which can lead to missed detections and false positives. Although it may be difficult to verify the completeness of vulnerability matching results after implementation, manual checks and other verification methods are expected to be employed to validate the results of the vulnerability matching process.

Activities for vulnerability identification (1) through (4) should be conducted regularly to respond to newly discovered vulnerabilities. When repeatedly performing vulnerability identification, managing information such as previously addressed vulnerabilities, their prioritization, and the association with newly discovered vulnerability information becomes a challenge. In the future, enhancements in VEX information and SBOM tool functionalities are expected to facilitate the efficient management of such vulnerability information.

7.4.2. Vulnerability Response Prioritization Phase

The vulnerability response prioritization phase consists of the following four steps:

- (1) Vulnerability filtering
- (2) Selection and acquisition of prioritization information
- (3) Category determination based on prioritization decision tree
- (4) Priority score evaluation

Key points for the specific implementation methods are outlined below for each step.

(1) Vulnerability filtering

Since the effort required to gather information externally and conduct comprehensive vulnerability response prioritization is significant, it is expected that

simple categorizations will be performed in advance for vulnerabilities that are easily identifiable. For instance, if a supplier has clearly stated that vulnerabilities in their software (components) have already been addressed, the prioritization step can be skipped, allowing for pre-categorization. In the future, if vendors begin to provide VEX information that includes details on whether a response to a vulnerability is necessary, this information could also facilitate filtering in this step, allowing for the omission of the prioritization step.

In the following steps, the method for prioritizing vulnerabilities—excluding those that do not require a response—by obtaining external information will be outlined.

(2) Selection and acquisition of prioritization information

In this step, the selection and acquisition of information necessary for prioritization will be conducted. The information needed for prioritization can be assessed using cost-effectiveness as a measure for vulnerability management through SBOM. The basic structure of cost-effectiveness can be understood as follows:

$\begin{aligned} \text{(Scale for vulnerability mapping)} &\propto \text{(Effect)} / \text{(Cost)} = \text{(Effect of reducing vulnerability risk)} / \text{(Cost)} \\ &\propto \text{(Possibility of threat)} * \text{(Vulnerability Residual Possibility)} * \text{(Impact)} / \text{(Cost)} \end{aligned}$

The effectiveness of using SBOM can be viewed in terms of security, specifically as the risk reduction effect achieved through vulnerability management³⁰. In the context of security, vulnerability risk is proportional to both the likelihood of an incident occurring and the severity of its impact (the magnitude of potential damage)^{31, 32}. The likelihood of an incident is influenced by both the probability of threats and the potential for vulnerabilities to remain unaddressed. Therefore, the cost-effectiveness can be understood in terms of this proportional relationship.

To compare and evaluate the magnitudes of these components, the following table presents relevant information:

³⁰ The effectiveness of SBOM encompasses not only security-related vulnerability risks but also risks associated with license compliance violations.

³¹ NISC presentation: "Cybersecurity Technology Issues from a Societal Perspective," Masaki Ishiguro, Mitsubishi Research Institute, 2019.

³² Information Processing Society of Japan Special Issue: "Cybersecurity in the Digital Economy Era," "Cybersecurity Economics," Masaki Ishiguro, Mitsubishi Research Institute, 2018.

Table 7-2 Information required for vulnerability response prioritization

Evaluation category			Evaluation item	Explanation and importance considerations
Risk	Occurrence probability	Threat occurrence probability (External factors)	Incident (Yes/No/Unknown)	There have been actual exploitations and incidents, indicating a high urgency.
			Public release of Exploit Code (Yes/No/Unknown)	Exploit code has been made public, increasing the likelihood of exploitation.
		Residual vulnerability probability (Internal factors)	VEX vulnerability status (Impact: Yes/No/Unknown)	This assessment has been conducted directly by developers utilizing the vulnerable components, ensuring high accuracy.
			Independent assessment of exploitability (Exploitable: Yes/No/Unknown)	If VEX cannot be obtained, an independent assessment of exploitability will be conducted. Unlike VEX, which is created by the component developers, this assessment may have lower accuracy.
			Applicability of advisory mitigation measures (Applicable: Yes/No/Unknown)	The response measures for vulnerabilities are generally applicable; however, unless the component ID and vulnerability ID are completely matched, the accuracy of exploitability is not high.
			Availability of vulnerability fix patch (Zero-Day)	For vendors, the absence of a vulnerability fix patch increases their responsibility towards users and procurers, thus elevating the priority.
	Impact level	CVSS score (particularly Impact Assessment)		The assessment of impact and severity is based on general cases and is not tailored to user environments, which may result in lower accuracy.
		User impact assessment (Importance of information assets - CIA)		The evaluation focused on the user's information assets (CIA) is based on actual conditions and is expected to be highly accurate.

Evaluation category		Evaluation item	Explanation and importance considerations
			External services typically have a higher impact than internal systems (e.g., total values of CIA elements rated as 2, 1, 0).
		Impact on numerous products and services, High volume of Inquiries	There is a possibility of impact in the later stages (3-level assessment: 3, 2, 1).
Cost		Service interruption or degradation	Consider the impact of service interruptions or degradations, both internal and external, to determine appropriate timing (3, 2, 1).
		Software remediation	If the supplier's fixes are delayed, assess the timing for applying internal fixes.
		Impact testing of fixes / Implementation of fixes	Evaluate the feasibility of conducting impact testing for the applied fixes.
		Cost of exploitability assessment	Assess the cost of performing exploitability evaluations internally, as a substitute for the supplier, to inform decision-making.

By collecting this information, prioritization can be conducted based on the considerations outlined in the subsequent steps. These information sources mainly include vulnerability databases and SBOM tools, as indicated in Section 7.4.1(3). Companies can select and choose among these sources according to their policy requirements and available budget.

(3) Category determination based on prioritization decision tree

Based on the selection and acquisition of the necessary information for prioritization, a category determination for vulnerability response prioritization (priority sorting) will be conducted using the information obtained. To achieve this, it is important to utilize an internationally standardized framework to ensure accountability and global consistency. Therefore, the SSVC (Stakeholder-Specific Vulnerability Categorization)³³ framework proposed by the U.S. CISA will be employed. In SSVC, a decision tree is structured based on conditional branches concerning exploitability, automatability, technical impact, and mission & well-being. As a result, the categorization for prioritization is determined into four groups as below:

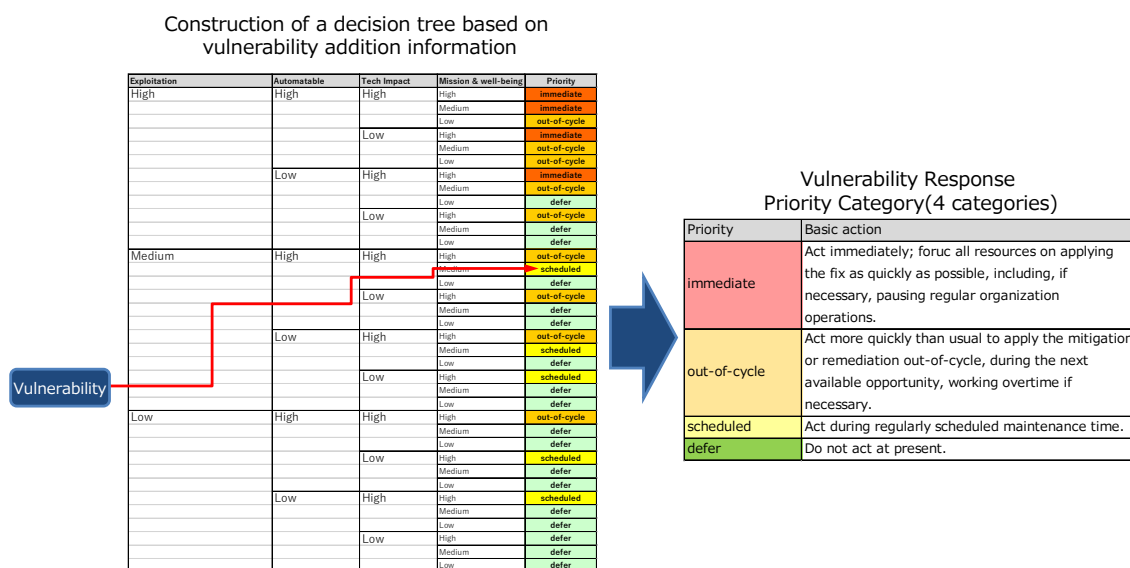


Figure 7-7 Structure of the decision tree for vulnerability response prioritization based on SSVC and categorization (Four categories)

The criteria for the decision-making branches can vary based on each company's

³³ CISA, SSVC (Stakeholder-Specific Vulnerability Categorization)
<https://www.cisa.gov/sites/default/files/publications/cisa-ssvc-guide%20508c.pdf>

security policy, allowing for some discretion. However, through evaluations conducted as part of the METI SBOM PoC project, the rationale has been organized. Each company is expected to refer to the principles outlined in this guide and determine specific criteria in line with their own policies.

The criteria for decision-making branches are expected to vary based on the roles and technical capabilities of companies. It is efficient and practical for the organizations that developed the software—such as equipment manufacturers and component suppliers—to handle vulnerability remediation for their own developed components. Therefore, the decision tree differentiates between development organizations and user organizations. Additionally, due to varying levels of technical expertise in utilizing SBOM and managing vulnerabilities, practical capabilities differ based on this expertise. As a result, the decision tree for prioritizing vulnerabilities is categorized into four groups based on roles (development organizations vs. user organizations) and levels of technical capability (high vs. low), providing reference examples for each perspective.

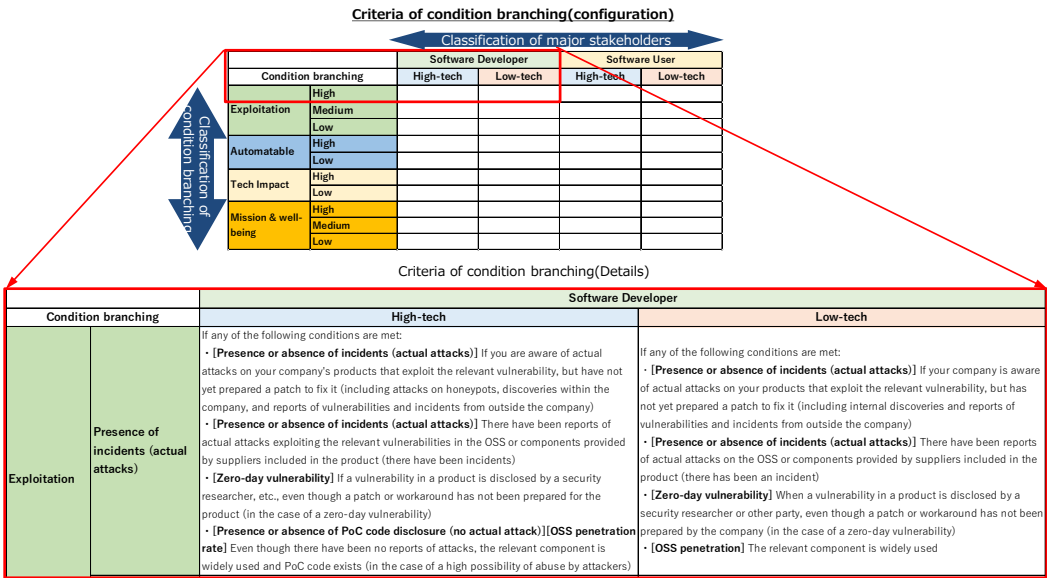


Figure 7-8 Overview of the criteria for vulnerability response prioritization judgment tree by four categories

The following table presents the judgment methods for each condition branch across the four categories. Table 7-3 illustrates specific methods for making prioritization category judgments based on the information used for vulnerability response prioritization shown in Table 7-2. Companies are expected to determine their judgment criteria based on their individual policies with reference to these examples.

Table 7-3 Method for prioritizing decisions by organizational category

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
Exploitation	High	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Presence or absence of incidents (actual attacks)] If you are aware of actual attacks on your company's products that exploit the relevant vulnerability, but have not yet prepared a patch to fix it (including attacks on honeypots, discoveries within the company, and reports of vulnerabilities and incidents from outside the company) • [Presence or absence of incidents (actual attacks)] There have been reports of actual attacks exploiting the relevant vulnerabilities in the OSS or components provided 	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Presence or absence of incidents (actual attacks)] If your company is aware of actual attacks on your products that exploit the relevant vulnerability, but has not yet prepared a patch to fix it (including internal discoveries and reports of vulnerabilities and incidents from outside the company) • [Presence or absence of incidents (actual attacks)] There have been reports of actual attacks on the OSS or components provided by suppliers included in the 	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Presence or absence of incidents (actual attacks)] If there are reports of actual attacks exploiting vulnerabilities in the product or OSS contained in the product (judged from the SBOM) • [Presence or absence of incidents (actual attacks)] If the product vendor reports that there is a high possibility of abuse • [Whether or not PoC code has been published (no actual attacks)] [OSS penetration rate] Although there have been no reports of attacks, the relevant product 	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Presence or absence of incidents (actual attacks)] There have been public announcements of actual cases of abuse by JPCERT/CC, news, or vendors • [Explanation of vulnerabilities] Vendors are recommending early application of fixes • [Presence or absence of incidents (actual attacks)] There have been reports from inside and outside the company about the possibility of attacks on the company's systems

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
		by suppliers included in the product (there have been incidents)	product (there has been an incident)	or component is widely used and PoC code exists (in cases where there is a high possibility of it being exploited by attackers)	
		<p>• [Zero-day vulnerability] If a vulnerability in a product is disclosed by a security researcher, etc., even though a patch or workaround has not been prepared for the product (in the case of a zero-day vulnerability)</p> <p>• [Presence or absence of PoC code disclosure (no actual attack)][OSS penetration rate] Even though there have been no reports of attacks, the relevant component is widely used and PoC code exists (in the case of a high possibility of abuse by attackers)</p>	<p>• [Zero-day vulnerability] When a vulnerability in a product is disclosed by a security researcher or other party, even though a patch or workaround has not been prepared by the company (in the case of a zero-day vulnerability)</p> <p>• [OSS penetration] The relevant component is widely used</p>	<p>• [Whether or not there have been incidents (actual attacks)] Reports have been received from inside and outside the company regarding the possibility of attacks on the company's systems</p>	

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
	Medium	<p>·[Whether or not PoC code is disclosed (without actual attacks)] If there is PoC code (exploit code) for vulnerabilities in the OSS or components provided by suppliers included in the product (however, this does not include cases where actual attacks have not been confirmed, or where PoC exists but it is a vulnerability discovered within the company and it has been confirmed that it has not been disclosed externally).</p>	<p>·[Whether or not PoC code is disclosed (without actual attacks)] If there is PoC code (exploit code) for vulnerabilities in the OSS or components provided by suppliers included in the product (however, this does not include cases where actual attacks have not been confirmed, or where PoC exists but it is a vulnerability discovered within the company and it has been confirmed that it has not been disclosed externally).</p>	<p>·[Whether or not PoC code is disclosed (without actual attack)] Although there have been no reports of actual attacks exploiting the vulnerability, if PoC code exists</p>	Other than those listed above.
	Low	<p>If any of the following conditions are met:</p> <p>·[Whether or not there was an incident (actual attack)][Whether or not the</p>	<p>If any of the following conditions are met:</p> <p>·[Whether or not an incident (actual attack) has occurred][Whether or not PoC code has been disclosed</p>	<p>·[Whether or not an incident (actual attack) has occurred][Whether or not PoC code has been disclosed (no actual attack)] If no</p>	

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
		<p>PoC code was disclosed (no actual attack)] If there are no attack examples or PoC code for the relevant vulnerability</p> <p>•[Whether or not there was an incident (actual attack)][Whether or not the PoC code was disclosed (no actual attack)] If the vulnerability was discovered within the company and there is no confirmed case of it being used for an attack externally</p>	<p>(no actual attack)]: In the case of the relevant vulnerability, neither attack examples nor PoC code exist</p> <p>•[Whether or not an incident (actual attack) has occurred][Whether or not PoC code has been disclosed (no actual attack)]: In the case of a vulnerability discovered within the company, there is no confirmed case of it being used for an attack externally</p>	<p>attack cases or PoC code have been discovered for the relevant vulnerability</p>	
Automatable	High	<p>If any of the following conditions are met.</p> <p>•[Location of the affected system] The affected vulnerability exists in a system that is located in a position that can be accessed from the Internet</p>	<p>If any of the following conditions are met:</p> <p>•[Location of the affected system] The affected vulnerability exists in a system that is accessible from the Internet</p>	<p>•[Location of the system in question] The vulnerability in question exists in a system that is accessible from the Internet</p> <p>Example: A publicly accessible web server, or a device (VPN, FW, etc.) that is located at the</p>	<p>•[Location of the system in question] The vulnerability in question exists in a system that is accessible from the Internet</p> <p>Example: A publicly accessible web server, or a device (VPN, FW, etc.) that is located at the</p>

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
		<p>Example: A publicly accessible web server, or a device (VPN, FW, etc.) that exists in a system that is the point of contact between an external network and an internal network</p> <p>•[Vulnerability description] RCE/Command Injection vulnerability</p>	<p>Example: A publicly accessible web server, or a device (VPN, FW, etc.) that exists in a system that is the point of contact between an external network and an internal network</p> <p>•[Vulnerability description] RCE/Command Injection vulnerability</p>	point of contact between an external network and an internal network	point of contact between an external network and an internal network
	Low	Other than those listed above.	Other than those listed above.	Other than those listed above.	Other than those listed above.
Tech Impact	High	<p>If any of the following conditions are met:</p> <p>•[Impact on system security functions] By using the relevant vulnerability, it is possible for an attacker to disable or bypass the security functions of the system (such as user authentication, access restrictions based on role</p>	Other than the following (including cases where the company is unable to judge the level of impact)	<p>If any of the following conditions are met:</p> <p>•[Impact on system security functions] If the vulnerability can be used to disable or bypass the security functions of the system (such as user authentication, access restrictions based on role settings, and tamper-proofing</p>	(This item is not evaluated, and is treated as all High)

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
		<p>settings, and tamper-proofing functions) that the target product has.</p> <p>•[Vulnerability description] By using the relevant vulnerability, it is possible for an attacker to obtain information contained in the target product.</p> <p>•[CVSS score] (only if the above judgment is difficult) CVSS score is Critical or High</p>		<p>functions) of the product in question.</p> <p>•[Vulnerability description] If the vulnerability can be used to obtain information on the product in question. If it is difficult to make the above judgment, the following conditions must be met.</p> <p>•[CVSS score] The CVSS score of the product development company (final vendor) is Critical or High (if there is no information from the final vendor, the CVSS score of the OSS that includes the relevant vulnerability is Critical or High)</p>	
	Low	Other than those listed above.	• [CVSS score] The CVSS score of the OSS that is affected is Middle or Low	Other than those listed above.	

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
Mission & well-being	High	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Nature of the target system] The target product is a system that serves as a point of contact between an external network and an internal network (VPN equipment, FW, etc.) • [Nature of the target system] The target product is a medical device classified as Class II/III/IV • [Characteristics of the target system] The components of the company that are affected by the vulnerability are used by the company or by the final product development team of another company and are 	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Nature of the target system] The target product is a system that serves as a point of contact between an external network and an internal network (VPN equipment, FW, etc.) • [Nature of the target system] The target product is a medical device classified as Class II/III/IV • [Characteristics of the target system] The components of the company that are affected by the vulnerability are used by the company or by the final product development team of another company and are 	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Nature of the target system] The target product is a system that serves as a point of contact between an external network and an internal network (VPN equipment, FW, etc.) • [Nature of the target system] The target product is a medical device classified as Class II/III/IV • [Nature of the target system] The vulnerability in question exists in a system that is the point of contact between the external and internal networks (VPN equipment, FW, etc.) 	<p>If any of the following conditions are met:</p> <ul style="list-style-type: none"> • [Nature of the target system] The vulnerability exists in a product that handles information that would have a critical impact on the company or its employees if leaked (e.g. information of extremely high sensitivity) • [Nature of the target system] The stoppage of the system with the vulnerability would have a significant impact on the company's business (e.g. stoppage of work for 80% or more of the employees) • [Characteristics of the target system] The vulnerability in question exists

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
		<p>incorporated into the final product (libraries, frameworks, etc.)</p> <p>•[Number of inquiries] (only if information can be obtained from the company's PSIRT or support department, etc.)</p> <p>Affects many of the company's products and services, or has already received many inquiries</p> <p>•Does not meet the "Medium Impact" criteria</p>	<p>incorporated into the final product (libraries, frameworks, etc.)</p> <p>•[Number of inquiries] (only if information can be obtained from the company's PSIRT or support department, etc.)</p> <p>Affects many of the company's products and services, or has already received many inquiries</p> <p>•Does not meet the "Medium Impact" criteria</p>	<p>•[Nature of the target system] A failure or malfunction of the system in question could have a fatal impact on human mental and physical health and the environment</p>	<p>in a system that is the point of contact between the external and internal networks (VPN equipment, FW, etc.)</p> <p>•[Characteristics of the target system] A failure or malfunction of the system in question could have a fatal impact on human mental or physical health or the environment</p>
	Medium	<p>•[Nature of the target system] It is confirmed that the target product (excluding cases where the company is a supplier providing components to the final product development vendor) does not store, retain, or transfer data</p>	<p>•[Nature of the target system] It is confirmed that the target product (excluding cases where the company is a supplier providing components to the final product development vendor) does not store, retain, or transfer data</p>	<p>If any of the following conditions are met:</p> <p>•[Nature of the target system] The vulnerability in question exists in a product that handles the company's confidential information</p>	<p>If any of the following conditions are met:</p> <p>•[Nature of the target system] The vulnerability in question exists in a product that handles the company's confidential information</p>

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
		obtained from users, such as personal information, or data entered by users (including information from sensors).	obtained from users, such as personal information, or data entered by users (including information from sensors).	<p>•[Nature of the target system] The stoppage of the system with the vulnerability in question would have an unacceptable impact on the company's business (e.g. stopping the work of a specific department, or more than half of the employees)</p> <p>•[Characteristics of the target system] There is a possibility (or the possibility cannot be denied) that a failure or defect in the relevant system could have an unignorable impact on human mental or physical health or the environment.</p>	<p>•[Nature of the target system] The stoppage of the system with the vulnerability in question would have an unacceptable impact on the company's business (e.g. stopping work in a particular department, stopping work for more than half of the employees, etc.)</p> <p>•[Nature of the target system] There is a possibility (or the possibility cannot be denied) that a failure or defect in the relevant system could have an unignorable impact on human mental or physical health or the environment.</p>
	Low	(Choose from the above two)	(Choose from the above two)	• [Nature of the target system] Even if the system with the relevant vulnerability	• [Nature of the target system] Even if the system with the relevant vulnerability

		Software Developer		Software User	
Condition branching		High-tech	Low-tech	High-tech	Low-tech
				stops, there will only be a minor impact (or no impact) on the company's business.	is stopped, there will only be a minor impact (or no impact).

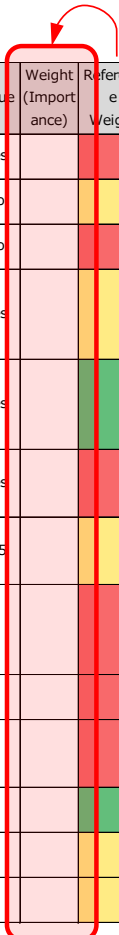
For each judgment node, it is also possible to refer to existing frameworks used by the company or industry frameworks.

Additionally, since prioritization may vary depending on the evaluator, it is recommended to verify the prioritization information based on the derivation process of individual prioritization data, the validity of assumptions and evaluation criteria, and the presence and content of the evidence that serves as the basis for the evaluation.

(4) Priority score evaluation

Alongside the categorization of vulnerability response prioritization in (3), a detailed prioritization can be conducted as needed through quantitative scoring within each category. This also involves organizing additional information that serves as evidence for the prioritization rationale. Score evaluation assigns numerical values (e.g., present, absent, unknown) to each evaluation criterion, and assesses them based on a weighted sum (weighted total).

Using reference weights as a base, adjustments can be made according to each organization's prioritization policy. By customizing these weights based on the circumstances and policies of each organization, it becomes possible to calculate scores for each vulnerability using the weighted total.



Evaluation category			Evaluation item	Value	Weight (Importance)	Reference Weight
Risk	Threat occurrence probability (External)		Incident (Yes/No/Unknown)	Yes		3
			Public release of Exploit Code (Yes/No/Unknown)	No		2
			VEX vulnerability status (Impact: Yes/No/Unknown)	No		3
	Occurrence probability	Residual vulnerability probability (Internal factors)	Independent assessment of exploitability (Exploitable: Yes/No/Unknown)	Yes		2
			Applicability of advisory mitigation measures (Applicable: Yes/No/Unknown)	Yes		1
			Availability of vulnerability fix patch (Zero-Day)	Yes		3
	Impact level		CVSS score (particularly Impact Assessment)	8.5		2
			User impact assessment (Importance of information)	3		3
			Impact on numerous products and services, High	2		3
	Cost		Service interruption or degradation	2		3
			Software remediation	1		1
			Impact testing of fixes / Implementation of fixes	2		2
			Cost of exploitability assessment	1		2

Figure 7-9 Evaluation criteria and reference weights for prioritization score assessment

It is anticipated that VEX information, which is expected to be utilized in prioritization and scoring, will be provided by vendors and others. However, since the availability of such information is currently limited, it is expected that the Known Exploited Vulnerabilities Catalog (KEVC)³⁴ provided by the U.S. CISA will be used when obtaining VEX is difficult. Although the Common Vulnerability Scoring System (CVSS) is relatively widely used, it does not consider the actual exploitation circumstances of vulnerabilities. Therefore, it may be useful to consider using the Exploit Prediction Scoring System (EPSS)³⁵, which was developed by the Forum of Incident Response and Security Teams (FIRST) as a complementary vulnerability assessment metric.

7.4.3. Information Sharing Phase

By extending the CISA SBOM Sharing Lifecycle³⁶, the methods for sharing information that includes not only SBOM but also vulnerability information and supplementary data will be organized.

Information sharing is expected to be considered based on the following two steps:

- (3.1) Identification of shared information and recipients
- (3.2) Identification and implementation of the sharing method

Below are examples of actions to be taken in these steps:

³⁴ Known Exploited Vulnerabilities Catalog <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>

³⁵ The EPSS Model, <https://www.first.org/epss/model>

³⁶ CISA, Software Bill of Materials (SBOM) Sharing Lifecycle Report <https://www.cisa.gov/resources-tools/resources/software-bill-materials-sbom-sharing-lifecycle-report>

Table 7-4 Steps in the Information Sharing Phase

Process, Phase, Step		Software Developer	Software User
(3.1) Identification of shared information and recipients	(3.1.1) Identification of shared information	<p>Identify the information to be shared (provided or acquired), such as vulnerability information obtained in Phase (1) of vulnerability management, additional information (exploitability, severity, advisory, VEX, etc.) obtained in Phase (2), and SBOM that have been fixed.</p> <p>In order to confirm the reproduction of vulnerabilities and implement fixes, it is also possible that developers will provide this information as additional information to prioritize and determine whether or not to address vulnerabilities.</p>	<p>It is also possible that users will request vendors to provide additional information on vulnerability response and prioritization by performing risk assessments based on usage conditions.</p>
	(3.1.2) Identification of recipients	<p>Identify the parties with whom the development and management departments (PSIRT, quality assurance) within the company and external parties (users, vendors) share information, and organize the information accordingly. Notifications are divided into push and pull types depending on the party providing and receiving the information.</p>	<p>Identify the vendors and customers (organizations using the service) with whom to share information, and prepare the information accordingly.</p> <p>Identify the contact points for the parties with whom to share information.</p>
	(3.1.3) Trigger of sharing (Discovery)	<p>It is expected that the developer will identify the vulnerability, prioritize it, and fix it in advance, so it is expected that the vendor will provide push notifications. However, prioritization will need to be</p>	<p>If SBOM has already been obtained, it is possible for the user organization to identify and prioritize vulnerabilities, and it is also possible for users to request that the vendor provide them (pull-type</p>

Process, Phase, Step		Software Developer	Software User
		determined according to the user's usage environment. It is expected that sharing will take place at the time the vulnerability is discovered and fixed.	notification). In some cases, the additional vulnerability information that can be obtained by users who are not developers may be limited.
(3.2) Identification and implementation of the sharing method	(3.2.1) Identification of the sharing method (Access)	<p>For SBOM and vulnerability information, the method of sharing is agreed upon between the parties sharing the information.</p> <p>For SBOM, there are two ways of sharing: manual, standalone tool file transfer and sharing, and SaaS sharing.</p> <p>As there is no standardization for vulnerability information and additional information other than CVSS, and SaaS support is limited, file transfer is expected.</p>	
	(3.2.2) Identification of the access privileges (Access, control)	<p>SBOM, vulnerability information confidentiality, and rights will be agreed upon, and disclosure categories (private/partially public/public) and access restrictions will be implemented as necessary.</p> <p>Agree on and implement management of disclosure categories, access restrictions, etc., depending on whether the tool is standalone or SaaS.</p> <p>Regarding sharing SBOM through the supply chain, SaaS is an effective method for efficiently sharing in real time with limited access, but it is necessary to evaluate the cost-effectiveness of SaaS costs.</p>	
	(3.2.3) Information sharing (Transport)	SBOM, vulnerability information, and additional information will be shared based on agreements on sharing methods, access permissions, etc.	

Process, Phase, Step		Software Developer	Software User
		<p>As shown in the CISA document [1], SBOM are divided into three levels of automation: manual transfer (e.g. email), partially automated, and fully automated based on standard rules. SaaS-type tools are used for automation.</p> <p>If SBOM have been shared in the past within the SW supply chain, vulnerabilities that have been identified as not requiring action as a result of vulnerability matching verification will also be notified, and action will be stopped across the entire supply chain.</p>	

7.4.4. Vulnerability Response Phase (Temporary and Fundamental Responses)

Vulnerability responses can be divided into fundamental responses involving fixes and temporary responses requiring immediate workarounds. Generally, it is expected that software developers will implement the fixes for vulnerabilities. Modifications to SBOM/VEX are necessary during fundamental responses. Various businesses often take on both software development and usage³⁷, and the balance of these roles may vary. Depending on whether an organization is in a development or usage position, a combination of the following processes will be required³⁸.

(4.1) Temporary Vulnerability Response

(4.2) Fundamental Vulnerability Response

Below are reference examples of actions to be taken during the Vulnerability Response Phase.

³⁷ Even service providers may have cases where the system department conducts development. Component suppliers may also use components further.

³⁸ If the procuring developer does not make fixes to the OSS software or SBOM, it is necessary to recognize in advance that the procurer is expected to respond equally to the developer.

Service Operator		
System Integrator		
Equipment Manufacturer		
Component Supplier		
Process, Phase, Step	Software Developer	Software User
(4.1) Temporary Vulnerability Response	(4.1.1) Consideration of provisional measures	<ul style="list-style-type: none"> Consideration of provisional measures before vulnerability fixes (suspension of use, reduced functionality, workarounds, etc.). Workarounds include a variety of measures, such as adding or changing protective mechanisms, changing settings, and restricting users. Consideration of provisional measures within the user organization (suspension of use, reduced operation, workarounds, etc.). Confirmation of provisional measures with the vendor.
	(4.1.2) Application of provisional measures	<ul style="list-style-type: none"> Informing the supply chain (including user organizations) of the interim measures. (no SBOM modifications) Compare the temporary measures within the user organization with the temporary measures proposed by the vendor, and make a decision. Consider the degree of impact of internal and external services when judging the temporary measures. (no SBOM modifications)
(4.2) Fundamental Vulnerability Response	(4.2.1) Implementation of fundamental measures	<ul style="list-style-type: none"> If the vulnerability is in a part developed in-house, fix the vulnerability. If the vulnerability is in a part developed by a supplier, request the supplier to fix it, and then apply the vulnerability fix to the part developed in-house. Request vendors to fix vulnerabilities. Apply vendor-supplied patches. For critical infrastructure services, set target values such as deadlines for applying fixes.
	(4.2.2) Update of SBOM and VEX	<ul style="list-style-type: none"> Update the SBOM in line with the vulnerability fix. Create and update the VEX in line with the vulnerability fix. VEX is modified in response to vulnerability fixes provided by the final vendor, and false positives are avoided in subsequent vulnerability management. (Normally, user organizations do not modify SBOMs because they do not modify software vulnerabilities.)
	(4.2.3) Sharing of SBOM and VEX	<ul style="list-style-type: none"> Share SBOM and VEX with suppliers. Manage SBOM provenance as necessary. Obtain updated SBOM and VEX from the vendor to fix vulnerabilities. Manage the history of SBOM as necessary.

*1 In some cases, the system department of the service operator will carry out development. In some cases, component suppliers will also use the parts further.

*2 If the developer of the procurement source does not modify the OSS software or SBOM, the procurement source needs to be aware in advance that they will be required to take the same measures as the developer.

Figure 7-10 Steps for vulnerability response and examples of implementation by stakeholder

Through the steps composed of the above phases, vulnerability management utilizing SBOM can be effectively carried out.

8. Appendix: SBOM Compliance Model

8.1. Purpose and background

8.1.1. Purpose

This chapter presents a method for visualizing the differences in the scope of SBOM generation and utilization. By using this method, software products with a high level of software management, such as vulnerability management, can be evaluated and selected in software transactions. This will enhance the incentives for suppliers to comply with SBOM requirements and promote the widespread adoption of SBOM, as well as standardize the levels of SBOM compliance.

While the Guidance (Chapters 1–6) outlines how to implement SBOM, this chapter aims to provide a framework for securing incentives regarding SBOM through the visualization of SBOM compliance levels. It addresses what actions should be taken (What) and why they should be taken (Why).

8.1.2. Awareness of issues

The utilization of SBOM is not a simple binary choice of whether to implement it or not; rather, it is determined by a variety of options based on the scope of identifying software components and the corresponding vulnerability management. There exist numerous levels of response. Identifying reusable components presents significant technical and cost challenges, and the extent to which components can be identified has a substantial impact on the level of vulnerability management. Additionally, costs and benefits vary greatly depending on the SBOM compliance level, making it crucial to assess the appropriate level of response according to the risks inherent in the specific field.

If there is no mechanism for visualizing the differences in SBOM compliance levels that allow for comparison between products, there will be little incentive to achieve an appropriate level of SBOM compliance. For instance, even if a high level of SBOM compliance is achieved at a significant cost, it won't lead to product selection based on this compliance unless software procurers recognize the value and comparability of that high level. Without such recognition, there is no incentive to invest in SBOM compliance. However, if the differences in SBOM compliance levels are visualized within a common framework that allows for comparison, it becomes possible to select products with high SBOM compliance in fields where robust vulnerability management is required, thereby promoting the optimization of SBOM

compliance levels according to the risks and requirements of those fields.

Until now, there has been no common framework available for comparing SBOM compliance levels, which has led to a lack of incentive to achieve an appropriate level of SBOM compliance. Consequently, it is necessary to provide a visualization framework that is comparable and universally applicable to enhance incentives for SBOM compliance. This framework should clearly delineate the scope of SBOM generation and utilization, as well as the corresponding compliance levels, thereby fostering greater motivation for organizations to adopt effective SBOM practices.

8.1.3. Target readers

The SBOM Compliance Model outlined in this chapter is aimed at both software and SBOM suppliers and procurers throughout the supply chain. It serves as a communication tool to visualize and agree on the security quality of software (such as configuration management and vulnerability management levels) during contracts between both parties. For software and SBOM suppliers, the target audience includes development and operations departments, as well as security teams (PSIRTs). For software procurers, the audience encompasses personnel from user companies, procurement departments, development teams, quality assurance departments, and security departments of development firms. Additionally, this model is important for executives and CISOs who are held accountable to society and business partners through its utilization. This chapter is designed for those who have understood the fundamentals of SBOM as presented in Chapters 1 through 6 and aims to provide guidance for utilizing this knowledge in software transactions and beyond.

8.1.4. Structure of this chapter

This chapter is organized as follows: In Section 8.1, the purpose and problem recognition are discussed, providing a concise summary of the key points. Section 8.2 summarizes the SBOM Compliance Model and the underlying SBOM visualization framework. Section 8.3 focuses on the positioning of the SBOM Compliance Model and its utilization methods. Finally, Sections 8.4 to 8.6 outline the legal frameworks and assumptions for each sector, present the sector specific SBOM Compliance Models organized based on the results of SBOM PoC, and discuss the usage methods and considerations for each sector.

8.2. SBOM visualization framework and Compliance Model

8.2.1. What is the SBOM Compliance Model?

The "SBOM Compliance Model" visualizes the scope of implementation expected for identifying components and managing vulnerabilities using SBOM. It indicates how far one should go in compliance, outlining both recommended and required items. Given the diverse elements of SBOM, the costs and effectiveness can vary significantly depending on the scope of compliance. Therefore, it is effective to aim for an appropriate scope of compliance based on the differing risks associated with various industrial sectors and system utilization environments. The SBOM Compliance Model serves to propose this appropriate scope of compliance.

For example, the scope of identifying software components using SBOM can vary significantly based on whether it includes components from contracted development sources or third-party vendors. Additionally, it can differ depending on whether the focus is solely on components directly used by those entities or if it also encompasses reused components. If the scope of component identification differs, the range of vulnerability detection will also vary accordingly, leading to significant implications.

8.2.2. Basic concepts and expected benefits

SBOM serves as an effective foundation for streamlining the management of software developed through the supply chain. By utilizing SBOM, it becomes possible to enhance the management of software based on component composition and vulnerability management, while also reducing risks associated with vulnerabilities. In the use of SBOM, it is crucial to visualize the differences in the extent of component identification and vulnerability management, rather than simply deciding whether to implement SBOM. The risks impacted by vulnerabilities can vary significantly based on the field and application, as well as the extent of SBOM's coverage, affecting the likelihood of residual vulnerabilities.

By visualizing the scope of SBOM compliance through a common, comparable framework and utilizing it in transactions, several benefits can be achieved. First, software suppliers and procurers can exchange not only the software itself but also the corresponding SBOM, including the scope of SBOM compliance and its level of adherence.

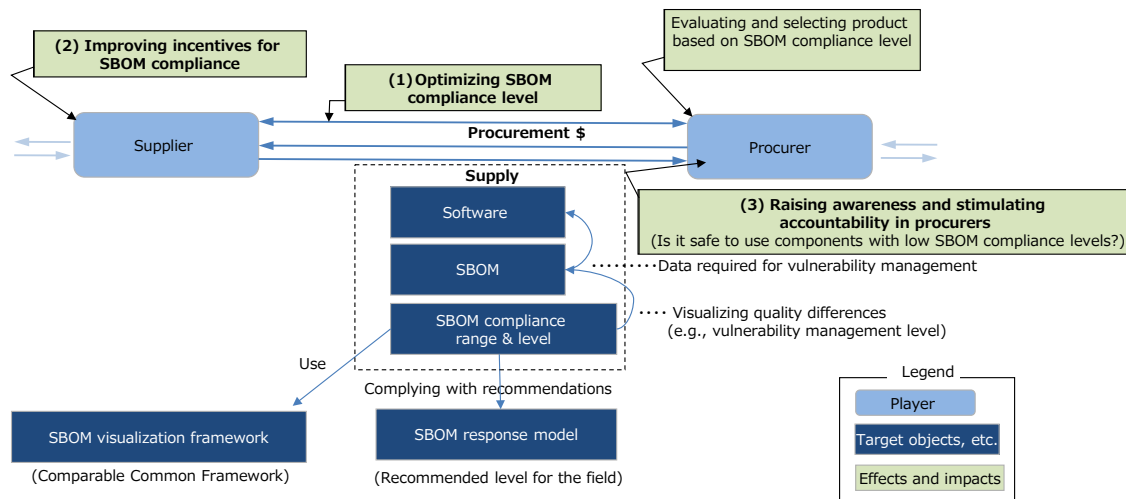


Figure 8-1 Relationships and benefits of users in the SBOM Compliance Model

Effects of visualizing the scope of SBOM compliance are as follows:

(i) Optimization of SBOM compliance level based on risk for both procurers and suppliers

Adjustments will be made to achieve an appropriate level of SBOM compliance based on the risks associated with different fields and applications. This involves balancing the costs reflected in product pricing with the scope of SBOM compliance (level of vulnerability management). For instance, products with low risks during use may require only minimal SBOM compliance, thereby reducing costs.

(ii) Enhanced incentives for suppliers regarding SBOM compliance.

By visualizing the scope of SBOM compliance, it becomes possible to demonstrate the high level of vulnerability management. This can enhance the procurer's evaluation and increase the perceived value of the product. Consequently, SBOM compliance becomes a means to enhance product value, leading to improved incentives for suppliers. However, it is important that the procurer's requirements for SBOM compliance do not impose excessive burdens on suppliers, ideally involving cost-sharing mechanisms.

(iii) Increased awareness and accountability among procurers

As verifying the Scope of SBOM compliance becomes standard practice during procurement, procurers will gain a greater awareness of SBOM compliance. By visualizing the Scope of SBOM compliance and understanding the level of vulnerability management, procurers will have more opportunities to fulfill their accountability to society and business partners. This enhanced awareness promotes informed decision-making and emphasizes the importance of security in

procurement processes.

The disparity between those who bear the costs of SBOM creation throughout the supply chain and those who benefit from vulnerability management using SBOM can hinder its widespread adoption if fair compensation is not established. The required level of SBOM varies by industry; for example, in critical infrastructure sectors where procurers demand high-level SBOM, suppliers must incur significant costs to produce them. If these suppliers do not receive adequate payment from procurers, their continued business viability may be jeopardized. Thus, promoting the adoption of SBOM involves challenges that cannot be resolved solely through advancements in technology or tools.

The visualization of the scope of SBOM compliance enhances incentives for both suppliers and procurers in transactions throughout the supply chain. Without a common framework for comparing SBOM compliance levels, such visualizations will not be utilized as comparison information during procurement, resulting in insufficient incentives for adopting higher-cost SBOM compliance levels.

8.2.3. SBOM visualization framework

(1) Structure of the visualization framework

The SBOM visualization framework serves as a structure to visualize the extent of its response options regarding the generation and utilization of SBOM. The effectiveness and cost of risk management for vulnerabilities through SBOM are determined by the range of components used in software and the scope of vulnerability management based on component information. Therefore, visualizing the response range of SBOM is crucial. This aims to provide a reference indicator for assessing how effectively software vulnerability risks are being addressed.

The framework for visualizing the response range of SBOM consists of two components: the various application categories related to SBOM generation and utilization, along with the corresponding response options and their status. The relationship between these elements is illustrated in the figure below.

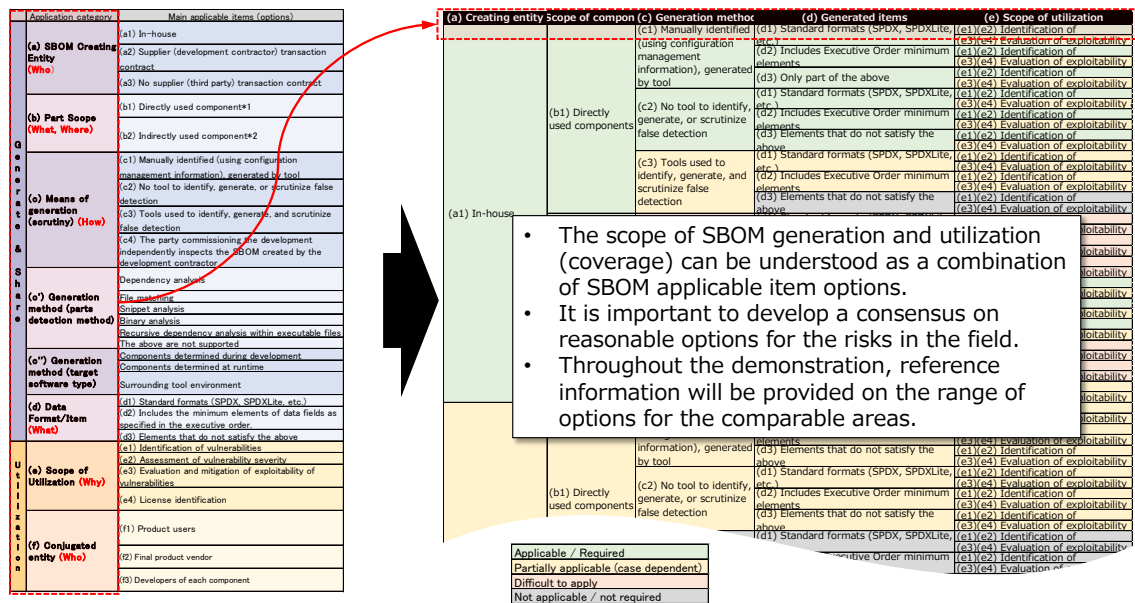


Figure 8-2 Visualization Framework for the scope of SBOM compliance (Conceptual image)

Below are detailed the specific SBOM compliance.

(2) Options for SBOM compliance items

The options for SBOM compliance items are organized based on the key choices that impact costs and effectiveness in each individual phase of SBOM generation and utilization. The overall response range and achievement level for configuration management and vulnerability management using SBOM can be identified as combinations of these options. The extraction and organization of the SBOM compliance item options were carried out using the following steps:

1) Extraction from key literature on SBOM

- NTIA: SBOM at a Glance³⁹
- NTIA: SBOM Options and Decision Points⁴⁰
- NTIA: Software Bill of Materials⁴¹

³⁹ https://ntia.gov/sites/default/files/publications/sbom_at_a_glance_apr2021_0.pdf

⁴⁰

https://ntia.gov/sites/default/files/publications/sbom_options_and_decision_points_20210427-1_0.pdf

⁴¹ <https://ntia.gov/page/software-bill-materials>

- CISA: Types of Software Bill of Material (SBOM) Documents⁴²
- CISA: Software Bill of Materials Site⁴³
- NIST: Software Security in Supply Chains: Software Bill of Materials (SBOM)⁴⁴
- SPDX: SPDX® Specification Version 2.3⁴⁵

2) Information extraction from domestic and international empirical results related to SBOM.

- NTIA: How-To Guidance for SBOM Generation in Healthcare⁴⁶
- METI: 2021 PoC Results

3) Organization of SBOM compliance Item Options (draft)

Based on the information from (1) and (2), the main response items related to SBOM creation and utilization were extracted, and the draft options for SBOM compliance items were organized.

4) Empirical evidence for SBOM by field

In the 2022 SBOM PoC conducted by the METI, feedback on the draft options for SBOM compliance items was gathered from companies in the medical device, automotive, and software product sectors, as well as participating industry associations. This feedback was used to revise the draft options for SBOM compliance items.

5) Feedback from the METI Software Task Force (2022, 7th to 9th Task Forces)

Opinions regarding SBOM from the Software Task Force were reviewed, along with feedback on any deficiencies or excesses in the draft options for SBOM compliance items.

6) Development of SBOM compliance item options (draft)

Based on the results from (1) to (5), the final draft of SBOM compliance item options was organized, reflecting the research and review outcomes up to FY 2022.

The draft of the SBOM compliance item options resulting from the above discussions is presented below. The options for response items in SBOM generation

⁴² <https://www.cisa.gov/sites/default/files/2023-04/sbom-types-document-508c.pdf>

⁴³ <https://www.cisa.gov/sbom>

⁴⁴ <https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity/software-security-supply-chains-software-1>

⁴⁵ <https://spdx.github.io/spdx-spec/v2.3/>

⁴⁶ https://ntia.gov/sites/default/files/publications/howto_guide_for_sbom_generation_v1_0.pdf

and utilization are structured in a 5W1H format to ensure comprehensiveness. In the following draft, the main application items (options) for SBOM classification are organized, and the costs associated with each item are categorized into three levels: High, Medium, and Low, along with explanations for each classification.

Table 8-1 Selection options of SBOM compliance item (Draft)

	Application category	Main applicable items (options)	Cost	Reasons for determining cost categories (main cost elements, assumptions, etc.)
G e n e r a t e & S h a r e	(a) SBOM Creating Entity (Who)	(a1) In-house	Low	Identify components directly used in in-house development from configuration files, etc., and generate SBOM. Including code modification components.
		(a2) Supplier (development contractor) transaction contract	Medium	Generate SBOMs for parts to be used in the software of contracted development companies with whom we have business contracts.
		(a3) No supplier (third party) transaction contract	High	SBOMs are created by OSS and off-the-shelf component vendors that are unable to make SBOMs a requirement through transaction agreements. (b2)(c2)
	(b) Part Scope (What, Where)	(b1) Directly used component*1	Low	The developer identifies the components to be used directly by the developer from configuration files, etc., and generates the SBOM with tools, etc.
		(b2) Indirectly used component*2	High	For third-party parts, generate SBOMs for recursively used parts.
	(c) Means of generation (scrutiny) (How)	(c1) Manually identified (using configuration management information), generated by tool	Low	Create component information to be used directly using configuration files, etc.
		(c2) No tool to identify, generate, or scrutinize false detection	Medium	Tools will be used to generate the SBOM and scrutiny will be omitted. The use of tools is assumed to be mainly for generating SBOMs for recursive parts, so commercial tools are assumed to be used.
		(c3) Tools used to identify, generate, and scrutinize false detection	High	Generate SBOMs using commercial tools, perform source code review, and scrutinize for false positives and omissions. (including recursive use components)
		(c4) The party commissioning the development independently inspects the SBOM created by the development contractor	High	When a development contractor accepts an SBOM created by a development contractor, the reliability of the SBOM is inspected by creating the SBOM independently with a tool or other means.
	(c') Generation method (parts detection method)	Dependency analysis	Medium	Static analysis of configuration information such as package manager.
		File matching	Medium	Detect file-by-file matches of source code using hash values. This includes detection of OSS libraries.
		Snippet analysis	High	Detect by partial string matching or similarity in the source code.
		Binary analysis	High	Similarity detection based on bit patterns in binary files.
		Recursive dependency analysis within executable files	High	For libraries already linked within the executable, recursively perform dependency analysis when building the library.
		The above are not supported	High	Convert pre-recognized parts to SBOM.
	(c'') Generation method (target software type)	Components determined during development	Low	Static libraries, applications
		Components determined at runtime	Medium	Runtime libraries, services (local, external cloud), OS, middleware, execution environment (container, VM, AP server)
		Surrounding tool environment	High	Tools used in development operations (installers, updaters, distribution packages, development environments, tool chains, SBOM tools)
	(d) Data Format/Item (What)	(d1) Standard formats (SPDX, SPDXLite, etc.)	Medium	Create in a standard format such as SPDX.
		(d2) Includes the minimum elements of data fields as specified in the executive order	Medium	Create an SBOM containing the minimum elements of the data fields in the Executive Order.
		(d3) Elements that do not satisfy the above	Low	Create your own minimum elements.
U t i l i z a t i o n	(e) Scope of Utilization (Why)	(e1) Identification of vulnerabilities	Low	Search and identify vulnerabilities in DBs such as NVD, JVN, etc.
		(e2) Assessment of vulnerability severity	Medium	Evaluate severity based on CVSS values and set priorities for vulnerability response.
		(e3) Evaluation and mitigation of exploitability of vulnerabilities	Medium	Evaluate the possibility of exploitation and the necessity of vulnerability countermeasures using VEX information, etc. Issue advisories on countermeasures, etc. as necessary.
		(e4) License identification	Medium	Identify the license and obtain the terms and conditions.
	(f) Conjugated entity (Who)	(f1) Product users	Low	If a vulnerability is identified, the use of the system is suspended and the company waits for the vendor to fix it. The damage is significant if business interruption costs are taken into account.
		(f2) Final product vendor	Medium	Notify users of vulnerabilities, request developers to correct them, and provide corrected builds and users. Report to authorities, ISAC, etc. as necessary.
		(f3) Developers of each component	High	The developer shall monitor and correct the vulnerabilities and provide the procurer with a corrected version. Report to the authorities, ISAC, etc. as necessary.

*Note 1: Directly used components: Components that are directly used by developers with whom there is a contractual relationship in the supply chain.

*Note 2: Indirectly used components: Components that are reused from components provided by suppliers (third parties) with whom there is no contractual relationship in the supply chain.

*Note 3: Costs are classified into three levels based on "Reasons for determining cost categories." The red text in the "Reasons for determining cost categories" column indicates areas that were revised during the PoC.

(3) Visualization of scope of SBOM compliance

The overall achievement level of vulnerability management and related areas is indicated according to the response range for SBOM generation and utilization. The scope of SBOM compliance can be visualized as combinations of the SBOM compliance item options. Although the number of all possible combinations can be substantial, major options can be extracted and visualized as combinations. An example is shown in the table below. The legend for the color coding used in the visualization is provided in Table 8-3. The options for each SBOM application category are not mutually exclusive; multiple selections are possible for the corresponding range. The scope of SBOM compliance is visualized and defined by the entirety of these tables. The metric that indicates the achievement level of the scope of SBOM compliance is termed the "SBOM compliance Level."

Table 8-2 Visualization and definition of the scope of SBOM compliance
(Example)

(a) Creating entity	(b) Scope of components	(c) Generation method	(d) Generated items	(e) Scope of utilization
(a1) In-house	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
	(b2) Indirectly used components	(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
(a2) Supplier (development contractor) transaction contract	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
	(b2) Indirectly uses components	(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
(a3) No supplier (third party) transaction contract	(b1) Developer itself	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
	(b2) Non-developer (procurers and users)	(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and (e3)(e4) Evaluation of exploitability and

The options for SBOM compliance items are visualized and color-coded according to the following four categories based on their response status:

**Table 8-3 Response categories for SBOM compliance item options
in the scope of SBOM compliance**

Compliance category	Color	Explanation of the status of response
Compliance	Green	Option is implemented or can be implemented
Partially compliance	Yellow	Option is partially implemented or can only be partially implemented
Difficult to comply	Orange	Option is not implemented or is difficult to implement
Unnecessary or excluded	Gray	No implementation required as it is addressed by the implementation of other items

8.3. SBOM Compliance Model and utilization methods

8.3.1. Positioning of the SBOM Compliance Model

The SBOM Compliance Model utilizes the SBOM visualization framework to illustrate a recommended or required response range based on the risks associated with different industrial sectors and applications. Risks and underlying legal frameworks vary by sector, and the effectiveness and costs of vulnerability management can significantly differ depending on the scope of SBOM compliance. Therefore, it is expected that SBOM Compliance Models will vary across sectors.

The figure below illustrates the relationship between the SBOM visualization framework and the SBOM Compliance Model.

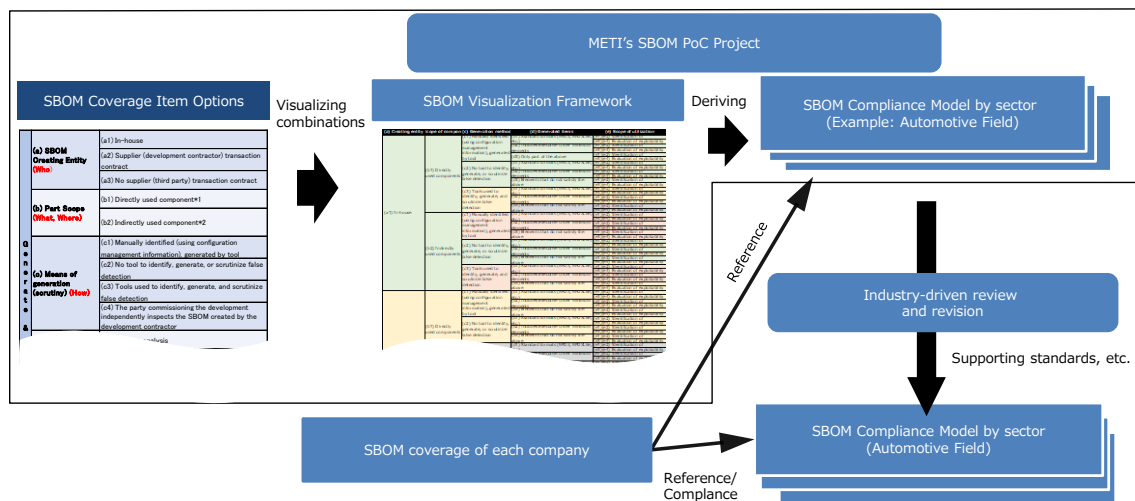


Figure 8-3 Relationship between SBOM Compliance Model and related elements

The SBOM visualization framework is a generic structure for visualizing the response range, including the scope of SBOM creation and utilization. The SBOM Compliance Model for each sector presents reference examples of the expected scope of SBOM compliance, considering the prerequisites, such as legal frameworks, and the feasibility based on the costs and benefits of SBOM compliance, as demonstrated through the METI's SBOM initiatives. The SBOM Compliance Models presented in this chapter do not guarantee compliance with regulatory requirements but aim to provide reference examples based on empirical results. It is anticipated that revisions of sector specific SBOM Compliance Models will occur through discussions and evaluations by regulatory authorities and industry stakeholders.

Using this framework, the METI's SBOM PoC evaluates costs and benefits based on prerequisites such as legal frameworks, transaction types, and development methods for each sector. The result is the "Sector-Specific SBOM Compliance Model (Reference Example)," which presents what is considered an appropriate SBOM application range for each sector. For non-regulatory areas, companies can refer to such SBOM Compliance Models to determine their own scope of SBOM compliance through the supply chain. All SBOM Compliance Models are required to comply with relevant legal frameworks and regulations. If a definitive industry version is developed, companies can reference these SBOM Compliance Models and, if necessary, add SBOM compliance items to enhance the value of their software and SBOM.

The SBOM visualization framework serves solely as a means to visualize the scope of SBOM compliance, and it is expected that the final scope of SBOM compliance

will be refined through discussions and evaluations specific to each industry.

8.3.2. Utilization methods

Based on Section 8.2.1, this section outlines the utilization methods for the SBOM visualization framework and the SBOM Compliance Model. The following figure illustrates the application methods and the impact of visualization in the procurement of software through the supply chain.

Software Procurement through the Supply Chain Market

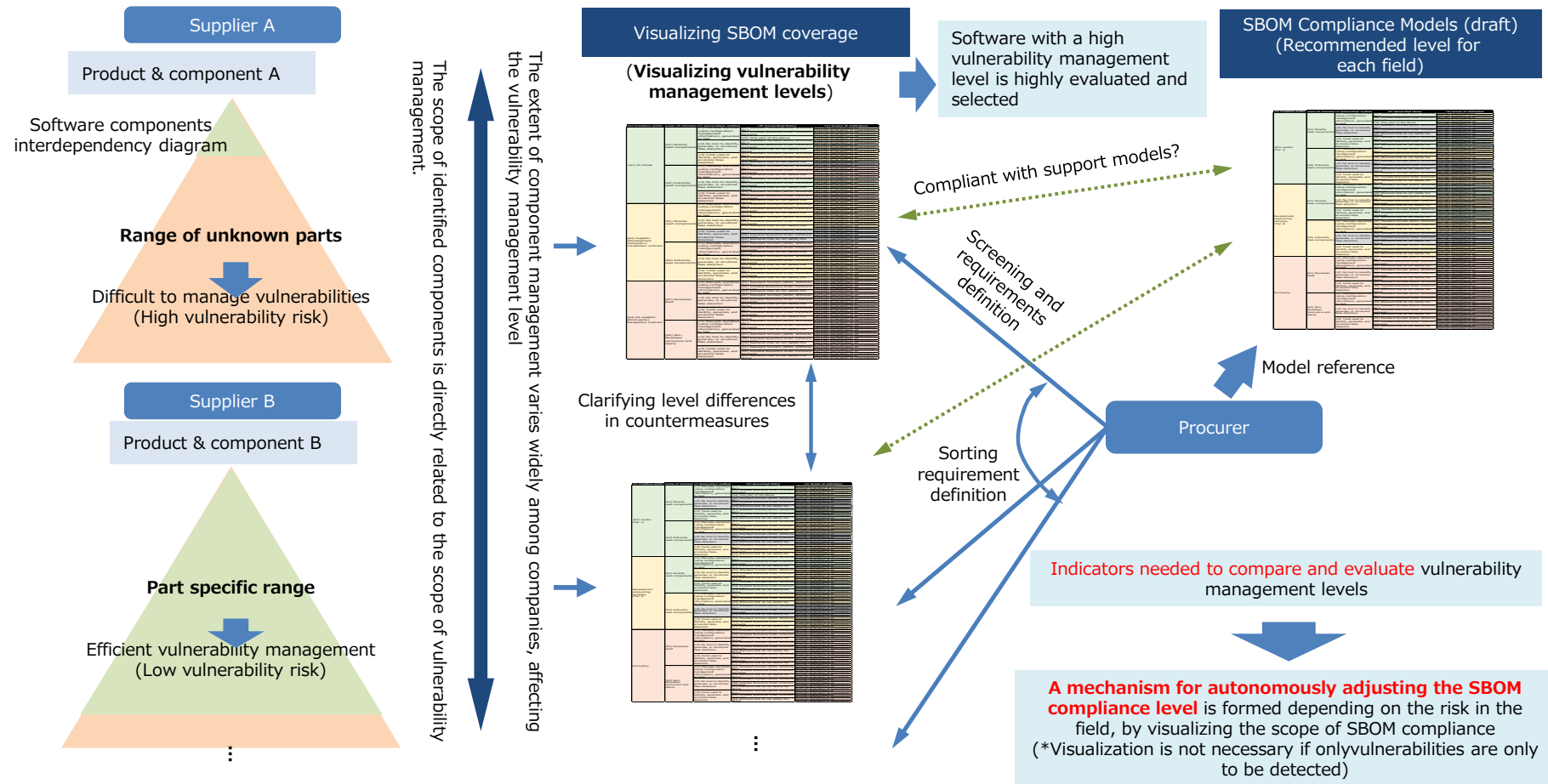


Figure 8-4 Enhancing the incentives for SBOM compliance through the visualization of the SBOM coverage (Conceptual image)

The figure illustrates the overall framework for promoting SBOM adoption through enhanced incentives, highlighting the relationship between suppliers and purchasers of products and components in software procurement.

The level of vulnerability management (vulnerability risk) for software provided by suppliers is determined by the extent of identifying the software's components and the scope of vulnerability management based on component information. The triangle shown on the supplier side of the figure represents an example of this concept, illustrating the differences in the extent of identifying components. The visualization of these differences in vulnerability management levels is depicted in the center of the figure, where the SBOM visualization framework is used to illustrate the scope of SBOM compliance. The green coverage rate in the visualized table of the scope of SBOM compliance can serve as a reference indicator for comparing and evaluating vulnerability risks.

The usage of the SBOM visualization framework differs based on whether software is procured through off-the-shelf sales or commissioned development. In the case of off-the-shelf sales, software is provided along with its SBOM, scope of SBOM compliance, and SBOM compliance level (the proportion of options addressed), allowing purchasers to make informed selections. Purchasers can compare the prices, functionalities, and SBOM compliance levels of similar software from multiple suppliers, leading to a preference for higher SBOM compliance levels that enhance the incentives for software providers to comply with SBOM requirements. Conversely, in commissioned development, it is conceivable that the purchaser will negotiate and agree on the coverage rate of the scope of SBOM compliance as a requirement with the development partner.

When a recommended or required range is established as the SBOM Compliance Model for a specific field, it is necessary to align the scope of SBOM compliance with that model. Additionally, even in sectors where an SBOM Compliance Model is defined, there are incentives to enhance value by providing software with additional SBOM compliance, leading to further improvements in incentives for software providers.

8.4. Reference example of SBOM Compliance Model (Automotive Sector)

In this chapter on sector specific SBOM Compliance Models, we will summarize the overview of legal frameworks and standards related to SBOM, the organized SBOM Compliance Model (Draft) based on empirical evidence, and key considerations for utilization.

8.4.1. Overview of legal frameworks and standards

In the automotive sector, the regulations developed by the United Nations Economic Commission for Europe's "World Forum for Harmonization of Vehicle Regulations (WP29)" in its subcommittee on "Automated Driving (GRVA)" are applicable as of July 2022. These include UN-R155 and UN-R156. While these regulations do not specify requirements related to SBOM, ISO/SAE 21434, referenced by the UN-R155 regulation required for type certification in regions such as Japan and Europe, includes examples of requirements such as software configuration management.

In the United States, the NHTSA released a draft guidance in early 2021 and conducted public comments. This guidance includes requirements for OEMs to create and maintain SBOM for software components used in ECUs and vehicles, indicating a future trend toward the recommendation of SBOM.

The relationship between automotive cybersecurity regulations, including those pertaining to SBOM, can be summarized as follows:

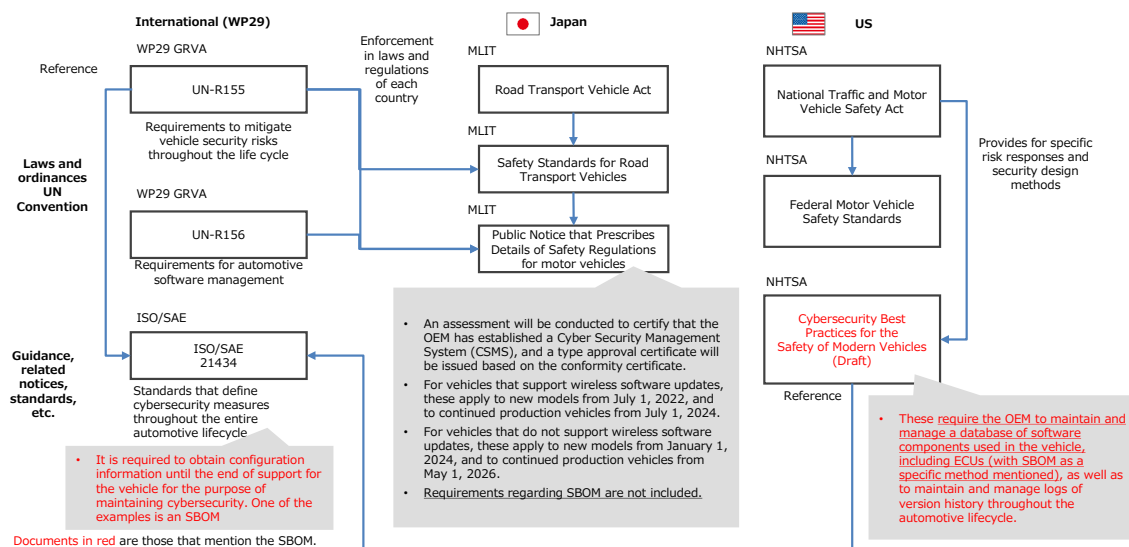


Figure 8-5 SBOM-related legal framework in the automotive sector

8.4.2. Proposed SBOM Compliance Model (Draft) based on the PoC

In the automotive sector, ISO/SAE 21434, referenced by the UN regulation UN-R155, includes software configuration information as an example of requirements for maintaining cybersecurity. Additionally, NHTSA's draft guidance released in early 2021 includes requirements for the creation and maintenance of SBOM. Therefore, it is expected that configuration management through SBOM will become increasingly necessary in the future.

In the 2022 SBOM PoC by the METI, the SBOM Compliance Model (reference example) was organized based on the following points. This model was specifically tailored for Tier 1 suppliers and ECU software suppliers (contracted development) involved in the PoC, considering the configuration management needs and technical feasibility required in the automotive sector.

The key points regarding the organization of the SBOM Compliance Model (Draft) are as follows:

- Based on the software configuration management requirements in the automotive cybersecurity standard ISO/SAE 21434, SBOM generation will be conducted, including for development contractors.
- For third-party suppliers, it is not mandatory to request SBOM due to challenges arising from contractual relationships.
- Since tool generated SBOM may contain false positives, thorough validation of identified components is necessary to fully realize the benefits of SBOM. Therefore, the SBOM generation method will include both manual identification

and items categorized as "identified/generated by tools with false positive validation."

- It was determined that indirect components can also be partially identified through manual methods, such as using package managers, leading to a partial application for these components.
- Tier 2 suppliers (ECU software suppliers) possess the relevant source code and hold information necessary for SBOM creation, like Tier 1 suppliers, allowing for SBOM generation under the same criteria.
- Both ECU vendors and ECU software suppliers (contracted development) face challenges in completely validating false positives for indirect components, so the partial application was adopted considering technical feasibility.
- For (e3)(e4) the evaluation of exploitability and severity, information provision has not progressed sufficiently, leading to a decision for partial application.

Based on the considerations, the organized SBOM Compliance Model (Draft), considering technical and cost feasibility, is as follows:

Table 8-4 The SBOM Compliance Model (reference example) for the automotive sector, examined through the PoC

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
ECU vendor (Tier 1)	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate,	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
		and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
	(b2) Indirectly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d3) Elements that do not satisfy the above	(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
(a2) Supplier (development contractor) transaction contract (Tier 2)	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
		scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
	(b2) Indirectly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d3) Elements that do not satisfy the above	(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
(a3) No supplier (third party) transaction contract	(b1) Developer itself	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
	(b2) Non-developer (procurers and users)	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d3) Elements that do not satisfy the above	(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

8.4.3. Utilization methods and considerations

As demonstrated in the previous sections, in the automotive field, software configuration management is cited as an example of requirements from the international standard ISO/SAE 21434, referenced by the international agreement regulation UNR-155, which is required for type certification safety standards. SBOM is expected to be utilized as an efficient means to meet these requirements. The SBOM Compliance Model can serve as a guide to the specific recommended scope of compliance items. The SBOM Compliance Model (Draft) organized during the PoC provides an example of a recommended scope, considering the needs and technical feasibility of companies in the automotive sector.

By referencing such examples, it is anticipated that stakeholders in the automotive industry (regulatory authorities, inspection agencies, automakers, suppliers, etc.) will further examine the SBOM Compliance Model and form a consensus within the industry. This collaborative effort is expected to provide specific recommendations for configuration management using SBOM, thereby promoting effective configuration and vulnerability management within the automotive sector.

8.5. Reference example of SBOM Compliance Model (Software Product Sector)

In this section on the SBOM Compliance Model for the software product sector, it will be summarized the overview of relevant laws and standards regarding SBOM, the proposed SBOM Compliance Model based on practical demonstrations, and key considerations for its utilization.

8.5.1. Overview of preconditions

Based on a U.S. presidential executive order, it is expected that the disclosure of SBOM will become mandatory in federal government software procurement by the end of 2022. Following the Executive Order issued in May 2021, several documents containing guidelines related to SBOM have been published in the U.S.

The executive order instructed the DHS to make recommendations regarding government procurement based on these guidelines by May 12, 2022. Following

this, a draft amendment to the FAR ⁴⁷ was announced in October 2023. The draft includes requirements for contractors supplying ICT products and services (e.g., telecommunications services, electronic media, IoT devices, operational technology) to create and maintain SBOM and grant government agencies access to them. Public comments on the FAR amendment will be accepted until February 2024, with a formal version expected to be released thereafter.

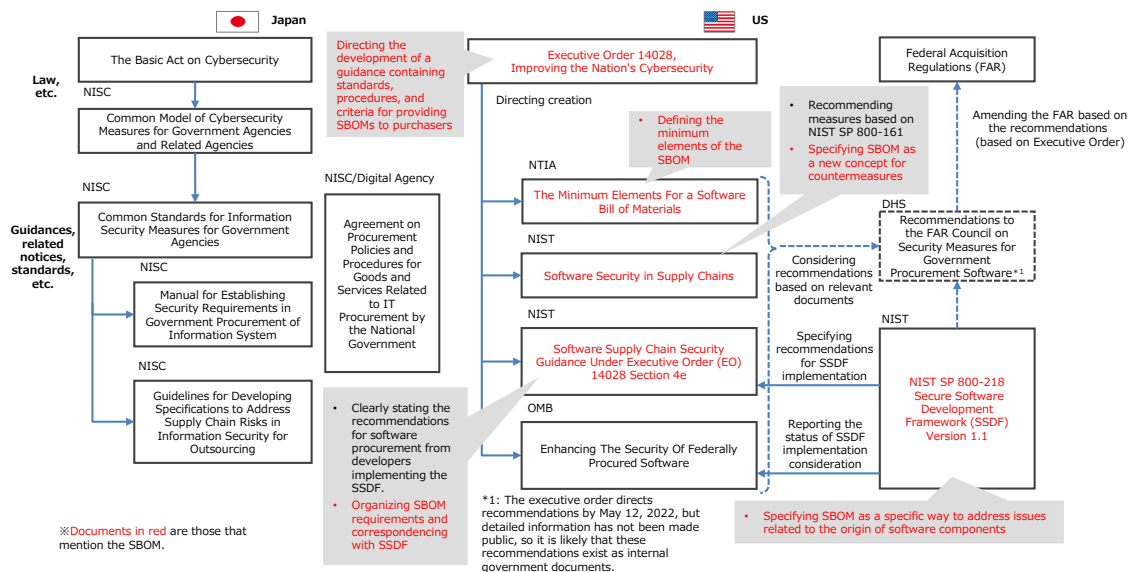


Figure 8-6 SBOM-related legal framework in the software product sector

The U.S. government procurement requirements will also impact Japanese companies supplying to the U.S. government. Additionally, from the perspective of international alignment, there is a possibility that the Japanese government and other countries may include SBOM-related requirements in their procurement standards. Therefore, addressing SBOM compliance. is considered important.

8.5.2. Proposed SBOM Compliance Model (Draft) based on the PoC

In the software product sector, using security software companies as examples, the METI 2022 SBOM PoC organized an SBOM Compliance Model (reference example) based on the following points. This model considers suppliers involved in the PoC, including software vendors and software service providers (contract

⁴⁷ Federal Acquisition Regulation: Cyber Threat and Incident Reporting and Information Sharing
<https://www.federalregister.gov/documents/2023/10/03/2023-21328/federal-acquisition-regulation-cyber-threat-and-incident-reporting-and-information-sharing>

development), while considering the configuration management and technical feasibility required in the software product sector.

The key points regarding the organization of the SBOM Compliance Model (proposal) are as follows:

- In the SBOM generation method "(c2) No tool to identify, generate, or scrutinize false detection," it is anticipated that tools will enhance work efficiency, particularly in identifying vulnerabilities. Given that the PoC companies have already incorporated SBOM creation and utilization as part of their development processes according to their security policies, this approach is deemed applicable.
- Regarding "(e3) (e4) Evaluation of exploitability and severity," useful information that can be included in the SBOM is currently almost unavailable. Therefore, it can only be partially applied, as manual verification and the combination of SBOM with external information are necessary for feasibility.
- Regarding "(c3) Tools used to identify, generate, and scrutinize false detection," for directly used components, the development team consciously utilizes these components, allowing for verification against an accurate list of direct components, thus enabling partial application. On the other hand, for indirectly used components, there is no efficient method to verify the "correct" information for comparison, and the examination requires significant time and effort. Considering feasibility, this approach is deemed difficult to apply.
- Regarding the examination of components detected by tools, it is considered that there are possibilities for both false positives and undetected items.
- The scope of the application has been identified based on the assumption of using a software composition analysis tool (paid) for component identification and SBOM generation.

Based on the considerations mentioned above, the organized SBOM Compliance Model (proposal) has been developed, taking into account technical and cost feasibility, as outlined below.

**Table 8-5 SBOM Compliance Model in the software product sector examined through practical validation
(Reference example)**

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
(a1) Final vendor	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d1) Standard formats (SPDX, SPDXLite, etc.)	
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
	(b2) Indirectly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
		scrutinize false detection		(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
(a2) Supplier (development contractor) transaction contract	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
(a3) No supplier (third party) transaction contract	(b1) Developer itself	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
			(d3) Elements that do not satisfy the above	(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
	(b2) Non-developer (procurers and users)	(c1) Manually identified (using configuration management information), generated by tool	(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Scope of utilization
		(c3) Tools used to identify, generate, and scrutinize false detection	(d3) Elements that do not satisfy the above	(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Elements that do not satisfy the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

8.5.3. Utilization methods and considerations

The software product field encompasses a variety of software types, including software procured for critical infrastructure and applications for individual entertainment. Currently, there are examples of SBOM being required as part of procurement standards under the SSDF for U.S. federal agencies; however, there are no specific regulations confirming the scope of SBOM compliance.

Given this context, in many areas of the software product field where no mandatory standards exist, promoting the practice of displaying and confirming the Scope of SBOM compliance between software buyers and sellers is expected to encourage autonomous adaptation to the necessary SBOM scope based on associated risks. If this practice becomes widespread, buyers will be able to understand the level of configuration and vulnerability management of the software they procure, thereby mitigating vulnerability risks. For suppliers, demonstrating a high level of SBOM compliance. can indicate robust vulnerability management, enhancing the product's value. Through these effects, it is anticipated that even without enforced standards, generalizing the display and confirmation of SBOM compliance. will lead to necessary adjustments according to the risks in the field.

8.6. Reference example of SBOM Compliance Model (Medical Device Sector)

This section provides an overview of a proposed SBOM Compliance Model specifically tailored for the medical device field, highlighting best practices and considerations unique to this sector.

8.6.1. Overview of legal framework and standards

In the medical device sector, consistent regulations are implemented for the manufacturing, sales, examination, certification, and post-market safety measures of medical devices, based on the Pharmaceutical and Medical Device Act (hereinafter referred to as "PMDA"). According to Article 41, Section 3 of the PMDA, the Minister of Health, Labour and Welfare can establish necessary standards to ensure the appropriateness of the characteristics, quality, and performance of medical devices, regenerative medical products, and in vitro diagnostic drugs, after

consulting the Pharmaceutical and Food Safety Council⁴⁸.

The Standard for Medical Devices Specified by the Minister of Health, Labour and Welfare pursuant to the provisions of Article 41, Paragraph 3 of the Act on Securing Quality, Efficacy and Safety of Products including Pharmaceuticals and Medical Devices (Ministry of Health, Labour and Welfare Notification No. 122 of 2005)⁴⁹, hereinafter referred to as the "Basic Requirements Standard,"⁵⁰ stipulates the fundamental requirements concerning the quality, efficacy, and safety that all medical devices and in vitro diagnostic drugs must possess.

In Article 12, Paragraph 2 of the Basic Requirements Standard, which addresses considerations for medical devices that use software, requirements for configuration management are outlined, and JIS T 2304 is specified as a standard for demonstrating conformity⁵¹. Furthermore, on March 9, 2023, the Ministry of Health, Labour and Welfare issued Notification No. 67⁵², which amended part of the standards for medical devices set by the Minister of Health, Labour and Welfare based on the provisions of Article 41, Paragraph 3 of the Act on Securing Quality, Efficacy, and Safety of Pharmaceuticals and Medical Devices. This amendment explicitly introduced cybersecurity requirements into Article 12, Paragraph 3 of the Basic Requirements Standard under the Pharmaceutical and Medical Device Act. The International Medical Device Regulators Forum (IMDRF) issued guidance in April 2020 aimed at achieving international harmonization of cybersecurity measures. As part of this international alignment, Japan is incorporating the IMDRF guidance into its Pharmaceutical and Medical Device Act regulations. Recently, two supplementary guidance documents were released: "IMDRF/CYBER WG/N73FINAL:2023 Principles and Practices for Software Bill of Materials for Medical Device Cybersecurity" (hereafter referred to as the IMDRF SBOM

⁴⁸ Pharmaceutical and Medical Device Act : <https://elaws.e-gov.go.jp/document?lawid=335AC0000000145>

⁴⁹ Standards for medical devices established by the Minister of Health, Labour and Welfare pursuant to the provisions of Article 41, Section 3 of the Act on Securing Quality, Efficacy and Safety of Products Including Pharmaceuticals and Medical Devices : https://www.mhlw.go.jp/web/t_doc?dataId=81aa6953&dataType=0&pageNo=1

⁵⁰ PMDA "Basic Requirements Standards" <https://www.pmda.go.jp/files/000240068.pdf>

⁵¹ Notification No. 0517-1, May 17, 2017, from the Director of the Medical Device Evaluation and Management Division, Pharmaceutical and Food Safety Bureau, Ministry of Health, Labour and Welfare https://www.std.pmda.go.jp/stdDB/Data/RefStd/Std_etc/H290517_0517-01_01.pdf

⁵² Official Gazette Main Edition, No. 933, March 9, 2023 <https://kanpou.npb.go.jp/20230309/20230309h00933/20230309h009330003f.html>

Guidance) and "IMDRF/CYBER WG/N70FINAL:2023 Principles and Practices for the Cybersecurity of Legacy Medical Devices" (hereafter referred to as the IMDRF Legacy Medical Device Guidance). Based on the content of these two supplementary guidance documents, the Medical Device Cybersecurity Working Group of the Japan Medical Device Industry Association (hereafter referred to as the "JMIA") has discussed handling Software Bill of Materials (SBOM), management of legacy medical devices, vulnerability remediation, and incident response. As a result, an updated version of the "Guide to Cybersecurity Implementation for Medical Devices (2nd Edition)" has been compiled for medical device manufacturers and distributors. Additionally, a guide for ensuring cybersecurity for medical devices in healthcare institutions has been developed, which outlines necessary measures and operational structures. To investigate trends in Japan, the report "Trends and Future Challenges in Medical Device Cybersecurity Regulations in Our Country"⁵³ was referenced to analyze the latest developments.

Organize the regulations related to pre-market security, including approval reviews, and add the findings regarding the consideration of domestic implementation of IMDRF guidance, trends in legislative amendments, etc., as illustrated in the following figure.

⁵³ Source: "Trends and Future Challenges in Medical Device Cybersecurity Regulations in Our Country," Journal of Medical Device Science, Vol. 90, No. 6 (2020). Refer to "Figure 1: Pre-Market Requirements for Medical Device Cybersecurity under the Pharmaceuticals and Medical Devices Act" and make some additions. https://www.jstage.jst.go.jp/article/jjmi/90/6/90_534/_article/-char/ja/

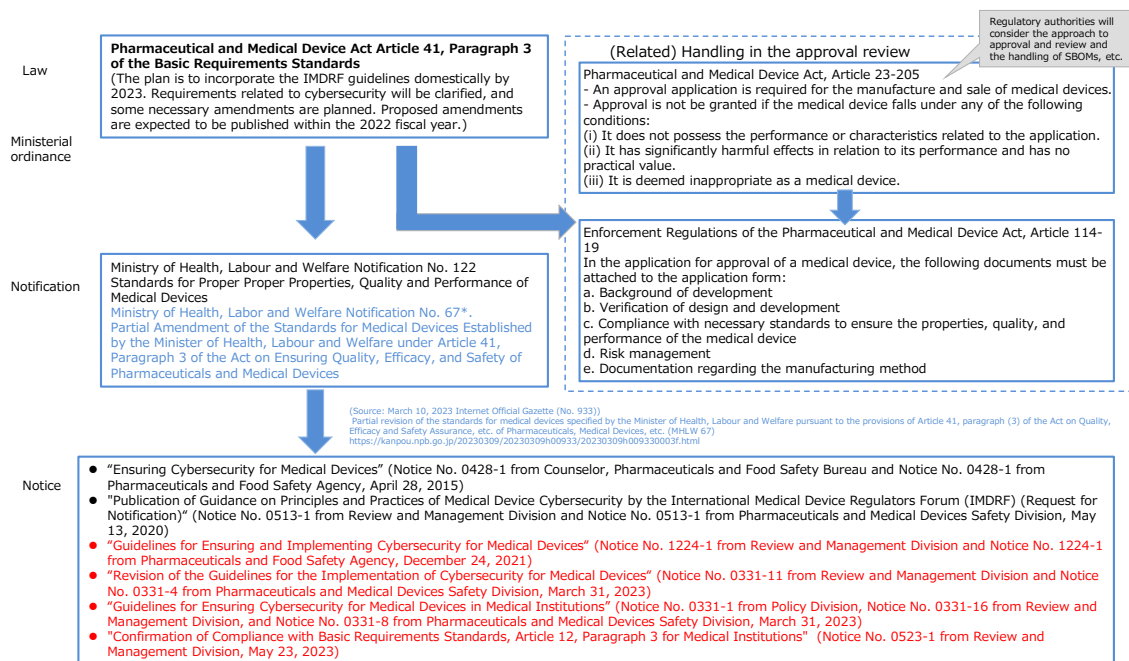


Figure 8-7 Overview of regulations related to pre-market security for medical devices

Source: Figure 1 in "Trends in Medical Device Cybersecurity Regulations in Japan and Future Challenges," *Medical Device Journal*, Vol. 90, No. 6 (2020) with some additions.

Additionally, I have organized the regulations related to post-market security for medical devices and added the findings regarding the consideration for the domestic implementation of IMDRF guidance. The following figure illustrates the post-market safety measures.

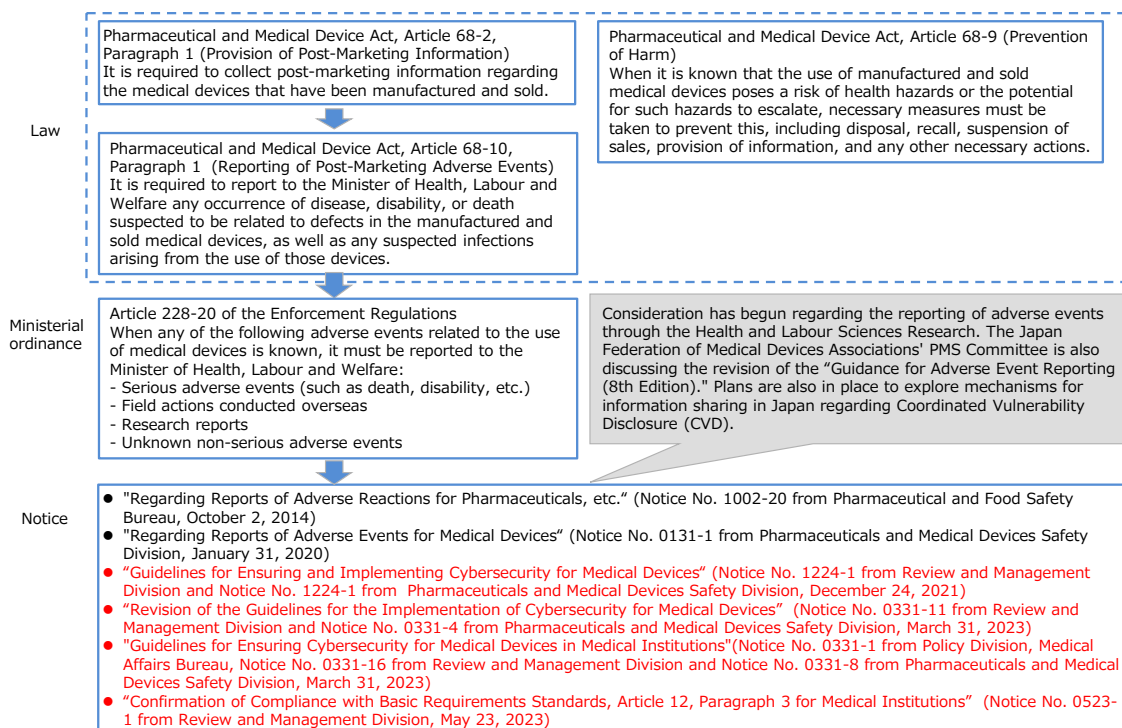


Figure 8-8 Overview of regulations related to post-market security for medical devices⁵⁴

Source: Figure 1 in "Trends in Medical Device Cybersecurity Regulations in Japan and Future Challenges," *Medical Device Journal*, Vol. 90, No. 6 (2020) with some additions.

In the field of medical devices, medical device manufacturers (hereafter referred to as "manufacturers") are already mandated to implement configuration management under Article 12, Paragraph 2 of the Basic Requirements Standard in the Pharmaceuticals and Medical Devices Act. Conformity can be confirmed through JIS T 2304. A new Paragraph 3 has been added to Article 12, clarifying the cybersecurity requirements. The standard related to Paragraph 3, JIS T 81001-5-1, has been established; it is a process standard for health software, including software incorporated into medical devices, and specifies activities that manufacturers must carry out as part of the development lifecycle. This standard can be used to confirm conformity with the requirements of Article 12, Paragraph 3.

SBOM is not required by JIS T 81001-5-1 or IEC 81001-5-1 (a standard for the medical device sector based on IEC 62443-4-1 from the control domain), but it is

⁵⁴ Regarding the considerations for post-market cybersecurity, the document titled 'Fundamental Approach to Reporting Malfunctions Related to Cybersecurity of Medical Devices' (issued on January 15, 2024, by the Pharmaceuticals and Medical Devices Agency, Notification No. 0115-2) has also been published.

required as customer-facing documentation in IEC TR 60601-4-5 (a standard for the medical device sector based on IEC 62443-4-2 from the control domain). With SBOM, customers can monitor the security-related risk environment and exchange information regarding security-related risks with manufacturers. An example of such information exchange includes security patches related to the software listed in the SBOM.

8.6.2. Proposed SBOM Compliance Model (Draft) based on the PoC

In the field of medical devices, the standard JIS T 81001-5-1 has been established to confirm conformity with the cybersecurity requirements of the Pharmaceuticals and Medical Devices Act. Additionally, as previously mentioned, guidance documents for medical device manufacturers and healthcare institutions have been developed based on IMDRF guidance and the IMDRF supplementary SBOM guidance. The guidance for medical device manufacturers requires risk management throughout the entire product lifecycle. The IMDRF guidance includes handling Software Bill of Materials (SBOM), management of legacy medical devices, vulnerability remediation, and incident response.

In the 2022 SBOM PoC conducted by the METI, the SBOM Compliance Model (reference example) was organized based on the following points. Here, we focus on Tier 1 development contractors, which are medical device manufacturers involved in the PoC, and consider the configuration management and technical feasibility required in the medical device sector to outline the SBOM Compliance Model (Draft).

The key points in organizing the SBOM Compliance Model (Draft) are as follows. It is important to note that the PoC results are case-dependent.

The SBOM Compliance Model (Draft) can serve as a reference when confirming whether accountability for at least the selected elements is fulfilled, particularly when using SBOM as one of the means to achieve configuration management throughout the lifecycle required by JIS T 2304, as well as the safety, efficacy, and security of health software and health IT systems under JIS T 81001-5-1.

- For the entity responsible for generating and sharing SBOM, "(a1) In-house" and "(a2) Supplier (development contractor) transaction contract," the scope of creation is deemed applicable by having the medical device manufacturers clearly identify the certification scope of the medical devices and the scope of

SBOM creation. It is necessary to require the contracted development partners to present the SBOM in the contract.

- In the case of SBOM generation and sharing by "(a3) No supplier (third party) transaction contract," some third parties may provide SBOM that can be verified by medical device manufacturers, allowing for partial applicability. However, there may be cases where the SBOM does not meet the eight elements outlined in the IMDRF supplementary SBOM guidance. Additionally, there may be instances where SBOM are not presented, making thorough examination of those SBOM difficult.
- Regarding the scope of SBOM creation and sharing for "(b1) Directly used components (components directly used by the development entity)," it is applicable because the developer can identify the components used directly from the configuration files, etc., by specifying the certification scope of the medical devices. Consequently, it is possible for the vendor of the contracted development partner to generate the SBOM using tools, making this approach applicable.
- Regarding the identification of SBOM creation and sharing scope for "(b2) Indirectly used components," it is possible to minimize detection omissions by using tools for source code analysis, binary analysis, and snippet analysis to detect components, and then manually cross-referencing the detected components. Therefore, it is desirable to use both tools and manual verification in combination. While verification is feasible, further consideration is needed regarding the level of scrutiny.
- Regarding the use of SBOM for "(e2) Vulnerability severity assessment," it is applicable as it targets the evaluation of CVSS scores using tools. For "(e3) Assessment and response to exploitability of vulnerabilities," it evaluates exploitability and the necessity of addressing vulnerabilities using VEX information and similar resources. It is necessary to issue advisories on mitigation strategies as needed; however, due to the difficulty of conducting thorough assessments, this was deemed not applicable in the PoC.
- In the PoC, the vulnerability management and response flow utilizing SBOM were examined. The medical device manufacturer, referred to as "(f2) Final product vendor," notifies users, such as healthcare institutions, of discovered vulnerabilities. This includes direct requests for modifications to the developers, as well as providing the modified build to users. A process for reporting to government agencies, regulatory authorities, and ISACs, as needed, was conceptually examined, making it applicable. However, due to a lack of

accumulated knowledge and difficulties in securing personnel and funding, it was determined that actual implementation would be challenging.

Table 8-3 SBOM Compliance Model examined in the medical device sector through PoC (Reference example)

Creating entity	Scope of components	Generation method	Generated items	Generated items
(a1) In-house	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXMLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXMLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
	(b2) Indirectly used components	(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXMLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXMLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

Creating entity	Scope of components	Generation method	Generated items	Generated items
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
(a2) Supplier (development contractor) transaction contract	(b1) Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
				(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Generated items
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
	(b2) Indirectly used components	(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses (e3)(e4) Evaluation of exploitability and severity
		(b1)		(e1)(e2) Identification of vulnerabilities and licenses

Creating entity	Scope of components	Generation method	Generated items	Generated items
(a3) No supplier (third party) transaction contract	Directly used components	(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c1) Manually identified (using configuration management information), generated by tool	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

Creating entity	Scope of components	Generation method	Generated items	Generated items
		(c2) No tool to identify, generate, or scrutinize false detection	(d1) Standard formats (SPDX, SPDXLite, etc.)	(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
		(c3) Tools used to identify, generate, and scrutinize false detection	(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d1) Standard formats (SPDX, SPDXLite, etc.)	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d2) Includes Executive Order minimum elements	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity
			(d3) Only part of the above	(e1)(e2) Identification of vulnerabilities and licenses
				(e3)(e4) Evaluation of exploitability and severity

8.6.3. Utilization methods and considerations

The SBOM Compliance Model (Draft) for the medical device sector presents a reference example based on the needs and technical feasibility confirmations by the PoC implementers in the medical device field, as well as advice from industry organizations such as the Japan Medical Device Industry Association (JMIA) considering domestic regulations. In the medical device PoC, guidance documents based on IMDRF guidance and supplementary IMDRF guidance have been provided as methodologies, and this content is followed accordingly.

8.7. Cross-sector comparison of the SBOM Compliance Models (Draft)

Regarding the selection options for SBOM-related items, a cross-sector comparison of the revised SBOM Compliance Model, which was evaluated and examined through the PoC, is organized and presented below.

In all sectors, obtaining SBOM from third parties, including open-source software (OSS), is challenging; therefore, responses through contracted development companies or tools are considered. In the medical device sector, SBOM generation and sharing must include contracted development partners, with the manufacturer bearing full responsibility. If the customer, such as a healthcare institution, requests the presentation of an SBOM, it is necessary to provide it and facilitate risk communication.

In the medical device sector, configuration management is legally mandated, so accountability for both directly and indirectly utilized components can be fulfilled. However, due to technical challenges in fully identifying indirect components related to OSS, only partial responses are implemented for software product sectors.

**Software product sector
(e.g., security software)**

Applicable / Compliant	Partially compliance (Case-dependent)	Difficult to comply	Unnecessary or excluded
------------------------	------------------------------------------	---------------------	----------------------------

Figure 8-9 Cross-sector comparison of SBOM Compliance Models (Draft)

9. Appendix: SBOM Contract Model

9.1. Background and purpose (problem awareness)

In the ordering and procurement of software components through the supply chain, there exists a cost burden on the side that creates the SBOM and a benefit for the side that obtains the SBOM to enhance vulnerability management. This creates asymmetry (bias) in the costs borne and the benefits received depending on the position of the parties involved. Therefore, to promote the widespread adoption of SBOM, it is crucial not only to outline the methods and procedures for implementation but also to ensure that, through contractual agreements, the suppliers who bear the costs of creating the SBOM are compensated by the procurers who benefit from it. Without this, it is anticipated that SBOM adoption will not reach an appropriate level. Thus, it is important to clarify the requirements and responsibilities regarding SBOM in the contracts and to stipulate payment for the corresponding costs.

This chapter aims to address the asymmetry in cost burdens and benefits between the parties involved in the ordering and procurement of SBOM, and to promote the widespread adoption of SBOM. It organizes the requirements, responsibilities, and cost-sharing matters related to SBOM that should be stipulated in contracts.

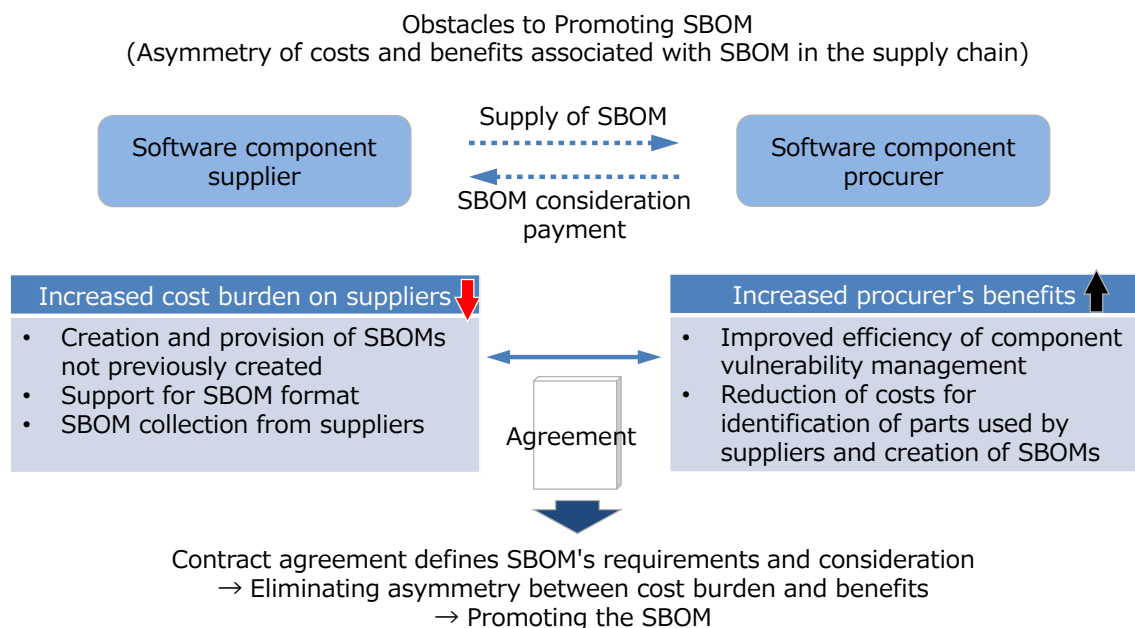


Figure 9-1 Obstacles to the promotion of SBOM adoption

9.2. Overview

9.2.1. What is the SBOM Contract Model?

The SBOM Contract Model is a reference example that outlines the main matters to be stipulated in contracts between software ordering and procurement parties regarding SBOM-related requirements, responsibilities, and cost burdens. It provides a conceptual framework for each company to create their contract clauses based on these reference examples.

9.2.2. Target readers

This chapter is primarily intended for legal professionals and developers involved in contracts that stipulate requirements, responsibilities, and cost burdens related to SBOM for software ordering and procurement parties.

9.2.3. Structure of this chapter

The structure of this chapter is as follows: Section 9.3 outlines the significance and utilization concepts of the SBOM Contract Model. Section 9.4 summarizes the components of the model and the matters to be stipulated. Section 9.5 discusses the relationship between the SBOM Compliance Model and the SBOM Contract Model. Section 9.6 describes the relationship between existing model contracts related to

software development and the SBOM Contract Model presented in this chapter. Section 9.7 illustrates the usage patterns and steps for the model. Finally, Section 9.8 discusses the status of the model and anticipated future revisions.

9.3. Concept of the Contract Model

The benefits of SBOM include the sharing of standardized component information through the supply chain and increased efficiency through automated processes. While the advantages for the commissioning party receiving the SBOM are significant, the contracted party may face additional burdens, resulting in varying benefits between the parties involved in the ordering and procurement.

To promote the adoption of SBOM through the supply chain, it is essential to establish agreements on cost burdens corresponding to the benefits received by each party. In the commissioning contracts, the scope of SBOM compliance., along with the associated cost burdens and responsibilities, must be clearly defined.

The SBOM Contract Model outlines the requirements for SBOM, along with the derived cost burdens and responsibilities. It serves as a reference for creating contracts tailored to each company and contributes to the effective utilization of SBOM.

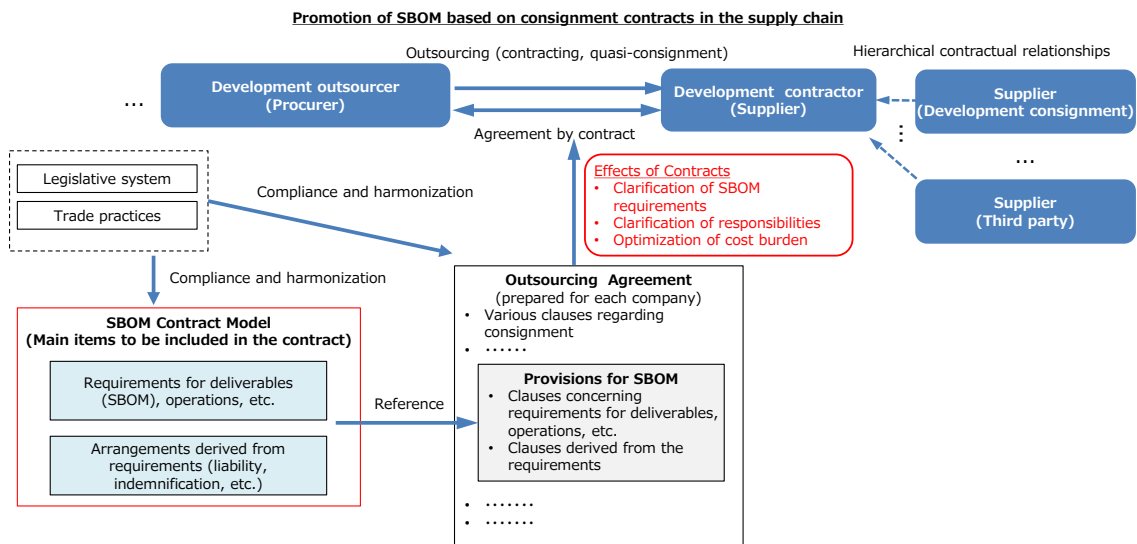


Figure 9-2 Concept of promoting SBOM adoption using the SBOM Contract Model

The model directly stipulates the relationship with companies that have contracted development agreements, but it can also indirectly define requirements regarding

the coverage of SBOM for third-party components. For example, the scope of SBOM creation can specify whether it includes contracted development partners or third parties (such as commercial off-the-shelf products or OSS), as well as whether it pertains to reusable components, thereby determining the coverage of third-party components.

9.4. SBOM Contract Model

9.4.1. Structure of the model

The SBOM Contract Model outlines the main matters that should be specified regarding SBOM in development contracts and order specifications. While it is necessary to ensure legal clarity in the actual provisions, this chapter emphasizes immediacy and organizes the key elements at the item level.

The structure of the model is based on the SBOM Compliance Model and includes requirements for the SBOM itself, as well as items related to responsibilities and guarantees from model contracts for software development published by organizations such as IPA and JEITA. Additionally, it focuses on key elements such as cost burdens for SBOM, rights, and confidentiality. The following structure has been organized accordingly.

Category		Outline
SBOM requirement	Format and standards	Establish requirements regarding the format and standards of SBOM.
	Quality and reliability	Establish requirements related to the quality of SBOM in the SBOM Compliance Model.
	Maintenance and operation	Establish requirements regarding vulnerability management and other related aspects in the SBOM Compliance Model.
Responsibility and warranty		Establish provisions regarding responsibility and liability for damages related to SBOM.
Cost burden		Establish provisions to ensure reasonable cost burdens associated with the creation and management of SBOM.
Rights and confidentiality		Establish provisions regarding intellectual property rights and related confidentiality concerning SBOM.

These elements have been organized considering the prerequisites specific to each field obtained through the METI's SBOM PoC project (refer to the appendix for legal systems, trading practices, and development environments). In the next section, the provisions of the SBOM Contract Model examined for each of these components will be presented.

9.4.2. Key provisions to be specified in the commissioning contract (Draft)

In response to the proposed structure of the SBOM Contract Model in the previous section, the main provisions to be specified in the commissioning contract have been organized based on the options from the SBOM Compliance Model and the prerequisites from the PoC (refer to the appendix for legal systems, trading practices, and development environments). These items have been revised based on feedback from the METI's Software Task Force.

Table 9-1 SBOM Contract Model (Key provisions to be specified in the development commissioning contract)

Category		Matters to be stipulated	Level
SBOM Requirements	Format Standard	(SBOM format)*1 Specify the SBOM standard format to be adopted. (Specify standards and versions of SPDX, CycloneDX, SWID, etc.)	Basic
		(ID standard)*1 Specify the part ID standard to be adopted. (CPE, PURL, SWD, proprietary format, etc.)	Basic
		(SBOM minimum elements)*1 Specify the minimum element among the element items of the SBOM format to be adopted, referring to the minimum element of the SBOM of NTIA.	Basic
	(Applicable to SBOM-compliant models)	(Supplier Contract Forms Covered) As the scope of SBOM creation, the scope by contract form of contract development agreement and third party terms and conditions (commercial off-the-shelf products, OSS) shall be specified.	Basic
		(Recursive use parts)*1 Specify whether direct use parts or recursive indirect use parts are included in the scope of SBOM creation.	Advanced
		(Scope of application of the composition analysis method)*1 For indirect use parts, the scope of application of the composition analysis method used to identify the parts is specified. (Dependency analysis, file matching, snippet analysis, etc.)	Advanced
		(Necessity of parts scrutiny)*1 Specifies whether or not manual scrutiny of false positives and omissions is required for the results of parts identification by the tool.	Advanced
		(Target phase of the component)*1 Specify the scope of the part information, such as build time, run time, cloud services, etc.	Advanced
		(Prior Agreement for Third Party Parts) When using third-party components (commercial components, OSS), this section defines whether or not prior declaration and agreement are required.	Basic
		(Sharing method)*1 This section defines real-time sharing by transfer by SBOM file or by SaaS, etc.	Basic
		(VEX support)*1 Specify whether to provide VEX information based on exploitability for vulnerability information related to SBOM.	Advanced
	Maintenance and Operation	(SBOM update)*1 Defines the deadline and frequency of updating the SBOM in response to software updates, SBOM defect fixes, etc.	Basic
		(Vulnerability Monitoring and Notification) During the operational phase of the software, monitor for vulnerabilities and stipulate a deadline for notification to the procurer when vulnerabilities are discovered.	Advanced
		(Vulnerability Response and Prioritization)*1 Specify whether or not information is to be provided to procurers regarding the need for vulnerability response and prioritization (triage) when vulnerabilities are discovered.	Advanced
		(EOL and EOS) This section defines the EOL and EOS for third party parts and contracted development parts and the notification of changes to their deadlines.	Advanced
		(Submission of Evidence) Specifies whether or not to require submission of evidence and third-party certification to prove conformity with SBOM requirements.	Advanced
		(Contract Nonconformity Liability) When nonconformity to SBOM requirements is found, it defines the necessity of defects response such as SBOM correction.	Basic
Liability and Warranty		(Compensation for damages)*2 Provide for the maximum amount of damages, etc., in the event of an accident caused by nonconformity with SBOM requirements. Includes damages for license violation.	Basic
		(Indemnification) For cases where evidence of conformance to SBOM requirements has been submitted, this section defines the limitations and disclaimers of liability for damages in the event that damages occur due to reasons attributable to technical limitations (e.g., false detection of tools).	Advanced
		(Quotation)*2 Prepare an estimate based on SBOM requirements, responsibilities and warranties, and stipulate the payment of consideration based on the agreed amount.	Basic
Cost Burden		(Attribution of Intellectual Property Rights) This section defines the intellectual property rights of the created SBOM, the ownership of the right to use the SBOM, and whether or not the SBOM can be provided to a third party.	Advanced
		(Confidentiality) This section defines the confidentiality and management of the SBOM and the prohibition of reverse engineering using the SBOM.	Advanced
Rights and Confidentiality			

*1. It is anticipated that this may also be included in the order specifications.

*2. It is expected that this will be standardized with general software development contracts.

The provisions in the SBOM Contract Model can be seen as the articulation of important requirements for vulnerability management and software quality assurance. These provisions are primarily expected to apply to contracts after the requirements definition phase.

9.5. Relationship and positioning of the SBOM Compliance Model and the SBOM Contract Model

The SBOM Compliance Model and the SBOM Contract Model are closely related, and it is anticipated that by utilizing one or both according to their respective purposes, the social implementation of SBOM can be advanced. The relationship and positioning of the SBOM Compliance Model and the SBOM Contract Model are illustrated in the figure below.

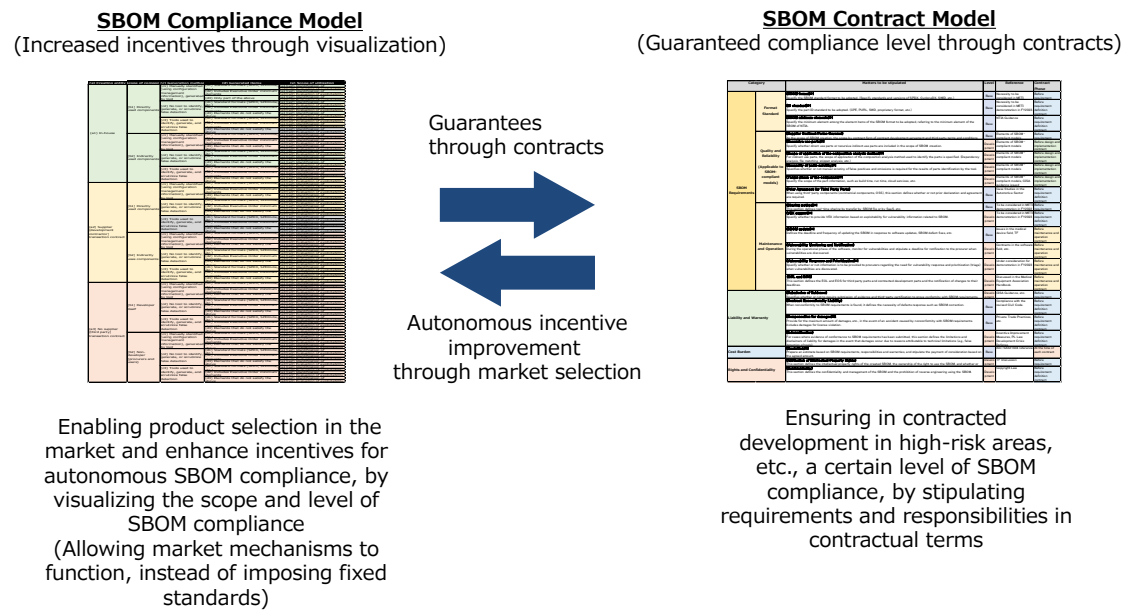


Figure 9-2 Relationship and positioning of the SBOM Compliance Model and the SBOM Contract Model (Overview)

The SBOM Compliance Model is a framework for visualizing the scope of SBOM compliance. By making the scope and level of SBOM compliance. visible, it aims to facilitate market product selection and enhance incentives for autonomous SBOM compliance. The scope of SBOM compliance. is not fixed by mandatory standards but is designed to allow market mechanisms to operate, enabling parties involved in software transactions to autonomously improve their SBOM compliance levels to

a reasonable standard.

On the other hand, the SBOM Contract Model ensures the level of SBOM compliance is guaranteed through contracts. In high-risk areas, such as outsourced development, the goal is to define requirements and responsibilities through contractual terms, thereby ensuring a certain level of SBOM compliance. By reaching an agreement through contract clauses, this approach promotes the social implementation of SBOM.

9.6. Relationship with existing model contracts

Model contracts for software play a crucial role in ensuring software quality, resolving discrepancies in understanding between ordering and supplying parties, and mitigating troubles that may arise during software development. Two representative model contracts are as follows:

- **Information-technology Promotion Agency, "Information System Model Transactions and Contracts (2nd Edition)"**

This document presents an ideal transaction and contract model aimed at enhancing the reliability of information systems and promoting transaction visibility.

- **Japan Electronics and Information Technology Industries Association (JEITA), "Explanation of the JEITA Software Development Model Contract"**

This document aims to clarify contract conditions to align the understanding of both users and vendors regarding cooperation, specifications, role distribution, and issues that need to be defined at the appropriate time, thereby contributing to the proper conduct of software development transactions and enhancing the reliability of information systems.

These model contracts provide comprehensive templates and explanations at the clause level for software development agreements. However, since these existing model contracts do not include provisions related to SBOM, the SBOM Contract Model can be utilized as a complement to them. That said, the SBOM Contract Model prioritizes immediacy in its publication, so it presents reference examples at the level of important matters rather than detailed templates at the clause level of the contracts.

9.7. Utilization patterns

There are two main patterns anticipated for utilizing the SBOM Contract Model.

1. **Based on in-house software development contracts**

Utilize an in-house software development contract template or an existing contract example as a base and incorporate the provisions of the SBOM Contract Model to finalize the contract.

(1) Evaluation of SBOM Contract Model provisions

Select provisions from the model based on the company's risk factors and required standards to determine which to adopt.

(2) Mapping with the SBOM Contract Model

Identify the relevant sections of the selected model provisions based on the structure of your company's contract template.

(3) Drafting clause with the model in a company contract

Consider the positioning and impact of the model provisions within your company's contract and draft the clause text accordingly.

(4) Discussion with the transaction partner

Engage in discussions with the partner regarding the draft contract reflecting the SBOM Contract Model, aiming to reach an agreement and finalize the contract including the SBOM provisions.

2. **Based on existing software development model contracts**

Utilize existing software development model contracts from IPA or JEITA as a foundation, incorporating the provisions of the SBOM Contract Model to complete the contract.

(1) Evaluation of SBOM Contract Model provisions

Select the provisions from the model based on your company's risks and requirements.

(2) Mapping with the SBOM Contract Model

Identify the relevant sections of the selected provisions in the SBOM Contract Model based on the structure of your software development model contract.

(3) Drafting clause with the model in a company contract

Consider the positioning and impact of the provisions from the model in your software development model contract and draft clause proposals accordingly.

(4) Discussion with the transaction partner

Discuss and reach an agreement with the transaction partner on the draft

contract that reflects the SBOM Contract Model, thereby completing the contract that includes the SBOM Contract Model.

Choose one of the utilization patterns outlined above and create a contract that corresponds to the SBOM Contract Model, following the steps of the selected utilization pattern.

9.8. Challenges and future directions for consideration

The SBOM Contract Model has been published with a priority on immediacy, and several challenges exist, as outlined below. Addressing these challenges in the future is expected to lead to revisions that make the model easier to implement.

- **Contractualization of the SBOM Contract Model provisions**

The SBOM Contract Model does not specify contract clauses; instead, it organizes important provisions that should be included in contracts. By providing legally clear draft clauses, this can facilitate a more user-friendly model. Users of the SBOM Contract Model can integrate existing general software development contract templates with the proposed clauses from the SBOM Contract Model to develop comprehensive contract templates.

- **Explanatory document for the SBOM Contract Model**

Users of the SBOM Contract Model are expected to create contract templates that integrate SBOM contractual clauses into software development model contracts and provide explanations for the use of each clause. Particularly, there is an expectation to summarize methods for selecting contract clauses based on the specific field and required levels, as well as the underlying rationale.

- **Distinction between contracts and specifications**

The provisions of the SBOM Contract Model can be categorized into matters that should be included in contracts and those that should be included in order specifications. It is anticipated that there will be guidance on how to utilize the SBOM Contract Model by breaking it down into such documents according to the nature of the contract, whether it be a work contract or a quasi-mandate contract.

10. Appendix

10.1. Checklist of actions for the introduction of SBOM

The following checklist summarizes the items to be implemented in the three phases of SBOM introduction: the environmental and system development phase, the SBOM production and sharing phase, and the SBOM use and management phase.

Table 10-1 Checklist of actions for the introduction of SBOM

Phase	Step	Actions for the introduction of SBOM	Check
Environment and system development phase	Clarification the scope of the SBOM application	Clarify information about the target software, such as information about development language, component type, development tools, etc.	<input type="checkbox"/>
		Create an accurate configuration diagram of the target software and visualize the target of the SBOM application.	<input type="checkbox"/>
		Clarify the contractual form and business practices with users and suppliers of the subject software.	<input type="checkbox"/>
		Confirm regulations and requirements for SBOM regarding the target software.	<input type="checkbox"/>
		Clarify the constraints within the organization (e.g., system constraints, cost constraints) regarding the introduction of SBOM.	<input type="checkbox"/>
		Clarify the scope of the SBOM application 5W1H (Five Ws and How) based on the organized information. For details, also refer to the SBOM Compliance Model in Section 8 (Appendix).	<input type="checkbox"/>
		Regarding the software to be procured or supplied, the requirements and responsibilities regarding SBOM will be clarified with the trading partner based on the SBOM Contract Model.	<input type="checkbox"/>

Phase	Step	Actions for the introduction of SBOM	Check
	SBOM tools selection	Organize the viewpoints for the selection of SBOM tools considering the development language of the target software and the constraints within the organization. (Examples of selection perspectives: functions, performance, analyzable information, analyzable data format, cost, supported formats, component analysis method, support systems, coordination with other tools, form of provision, user interface, operation method, supported software languages, Japanese support, etc.)	<input type="checkbox"/>
		Evaluate and select multiple SBOM tools based on the organized viewpoints.	<input type="checkbox"/>
	SBOM tools installation	Check the requirements of the environment where the SBOM tool can be installed and set up the environment. Consider a combination of tool functions and manual responses that meet the scope of support identified for the SBOM Compliance Model.	<input type="checkbox"/>
		Check the instruction manual and README file of the tool and then implement and configure an SBOM tool.	<input type="checkbox"/>
	Learning about SBOM tools	Learn how to use SBOM tools by checking the instruction manual and README file of the tool.	<input type="checkbox"/>
		Record know-how on how to use the tool and the outline of each function and share them within the organization.	<input type="checkbox"/>
SBOM production and sharing phase	Component analysis	Scan the target software and analyze the component information using an SBOM tool.	<input type="checkbox"/>
		Examine the analysis log of the SBOM tool and check whether the analysis has been correctly executed without any false positives or false negatives caused by errors or lack of information.	<input type="checkbox"/>

Phase	Step	Actions for the introduction of SBOM	Check
	SBOM production	Check the component analysis results to see if there are any false positives and false negatives.	<input type="checkbox"/>
		Determine the requirements for the SBOM to be produced, such as items, format, and output file format.	<input type="checkbox"/>
		Produce an SBOM that satisfies the requirements, by using the SBOM tool.	<input type="checkbox"/>
		In producing an SBOM, it is necessary to clarify who will do what, and to what extent throughout the supply chain, and to reach an agreement between the parties involved. Refer to Section 8 (Appendix).	<input type="checkbox"/>
	SBOM sharing	Share an SBOM with the users and/or suppliers of the target software as necessary after determining the method of sharing the SBOM.	<input type="checkbox"/>
		Consider using electronic signature technology or other technologies to prevent falsification of the sharing of SBOM data.	<input type="checkbox"/>
SBOM use and management phase	Vulnerability management, license management, etc.	Based on the output of the SBOM tool, assess the severity, evaluate the impact, fix the vulnerabilities, check the residual risk, and provide information to the relevant organizations.	<input type="checkbox"/>
		In the identification of vulnerability information, consider using (1) ready-made SBOM tools, (2) scripts that use the vulnerability DB API, and (3) the vulnerability DB WebUI, and select an appropriate method. (Refer to 7.4.1.)	<input type="checkbox"/>
		In the prioritization of vulnerability information, prioritize vulnerabilities based on cost-effectiveness, considering factors such as simple filtering of whether a vulnerability response is necessary, whether or not there have been incidents involving the vulnerability, whether or not exploit code	<input type="checkbox"/>

Phase	Step	Actions for the introduction of SBOM	Check
		has been released, the use of VEX information, and CVSS scores. In prioritization, classify priority categories as necessary, with reference to the SSSC approach. (Refer to 7.4.2.)	
		In the sharing of vulnerability information, identify the information to be shared, including additional information that is necessary for prioritizing vulnerability responses, the parties with whom the information is to be shared in the supply chain, and the means of sharing the information, and information is shared as necessary. (Refer to 7.4.3.)	<input type="checkbox"/>
		In responding to vulnerability information, implement both initial responses that do not involve fixing the vulnerability and fundamental responses that do involve fixing the vulnerability. (Refer to 7.4.4.)	<input type="checkbox"/>
		Based on the output of the SBOM tool, check whether there is any violation of the OSS license.	<input type="checkbox"/>
	SBOM information management	Keep the created SBOM for a certain period, including the change history, so that it can be referred to in case of inquiries from outside the company, etc.	<input type="checkbox"/>
		Manage the information contained in the SBOM and the SBOM itself appropriately.	<input type="checkbox"/>

10.2. Glossary

10.2.1. Terms related to SBOM and software

- **Attribute**
A characteristic or information about a component. In the case of SBOM in matrix format, it an attribute corresponds to a column.
- **Codebase**
The entire source code used to build a particular piece of software, application, component, etc.
- **Component**
A unit of software-defined by a supplier. A component is defined when it is built, packaged, or delivered by a supplier. Software products, equipment, libraries, and/or single files are also positioned as one component. An aggregation of components, such as OS, office suites, database systems, automobiles, automobile engine control units (ECU), medical image processing equipment, and installation packages, is also a component. Many components contain subcomponents.
- **Dependency Relationship**
A characterization of the relationship where software Y contains an upstream component X.
- **Element**
A part of the SBOM system.
- **Entity**
A company, association, organization, or individual associated with software or components.
- **EOL (End of Life)**
An expiration date is when a product or service is no longer sold or supported and should not be used continuously.
- **Intermediate Supplier**
A supplier that processes an upstream component into a new component for the downstream process. Many suppliers are treated as intermediate suppliers.
- **Minimum Elements**
The minimum elements to be included in an SBOM as announced by the NTIA

on July 12, 2021, based on Executive Order 14028 of the U.S. Specific definitions are provided based on three categories: data fields, automation support, and practices and processes.

- **OTS (Off-The-Shelf)**
A component of software that is commonly used by a supplier and for which the supplier cannot claim full software lifecycle management.
- **OSS (Open Source Software)**
A software whose source code has been made publicly available. Anyone is permitted to use, modify, and redistribute it.
- **Primary Component**
A target component described by the SBOM.
- **Proprietary Software**
A software whose intellectual property is retained by a software distributor and whose modification or reproduction is restricted.
- **Relationship Assertion**
An extent of one author's knowledge of another supplier's components. There are four categories: Unknown, Root/None, Partial, and Known.
- **Run-time Library**
A library required for program execution.
- **SBOM (Software Bill of Materials)**
An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships. These inventories should be comprehensive – or should explicitly state where they could not be. SBOM may include open source or proprietary software and can be widely available or access restricted.
- **SBOM Author**
An entity that creates an SBOM. When the author and supplier are different, this indicates that one entity (the author) is making claims about components created or included by a different entity (the supplier).
- **SBOM Consumer**
An entity that obtains SBOM. An entity can be both a supplier and consumer, using components with SBOM data in its own software, which is then passed downstream. An “end-user” consumer (that is not also a supplier) may also be called an operator or a leaf entity.

- **SBOM Entry**
An attribute related to a component of SBOM. In the case of a matrix SBOM, it corresponds to a row.
- **SBOM System**
A set of elements and processes that provide the ability to create, exchange, use, and manage SBOM.
- **SBOM Tool**
A tool to produce, share, utilize, or manage SBOM. An SBOM tool is also sometimes called an SBOM management tool, OSS management tool, or software configuration analysis (SCA) tool. In addition to a tool provided in a package, there are also tools provided as cloud software.
- **SCA (Software Composition Analysis)**
In a narrow sense, to identify the components used by the product. Generally, it is designed to manage vulnerabilities and license risks for each identified component.
- **Snippet**
A code fragment within a source code.
- **Subcomponent**
A component contained in a component.
- **Supplier**
An entity that develops, defines, and identifies a component, ideally an entity that creates an SBOM associated with that component. Suppliers are also called manufacturers, vendors, developers, system integrators, maintenance operators, and service providers. Most suppliers are also SBOM users. A supplier having no upstream components is also called a root entity.
- **Symbolic Link**
One of the functions in the OS file system, or another file indicating a specific file or directory.
- **Transitive Dependency**
A characterization of the relationship that if an upstream component X is included in software Y and component Z is included in component X then component Z is included in software Y.
- **VEX (Vulnerability Exploitability Exchange)**
A form of security advisory that indicates whether a particular product is

affected by a known vulnerability.

10.2.2. Other terms

- **Authentication**
Provision of assurance that a claimed characteristic of an entity is correct. [ISO/IEC 27000:2018]
- **Authorization**
To grant privileges, including the provision of access functions based on access privileges. [ISO 7498-2:1989]
- **CVSS (Common Vulnerability Scoring System)**
A rating method that allows quantitative comparison of the severity of vulnerabilities managed by FIRST (Forum of Incident Response and Security Teams) under the same criteria. The score is determined between 0.0 and 10.0.
- **CWE (Common Weakness Enumeration)**
A common standard for identifying types of security weaknesses (vulnerabilities) in software. The specifications were developed mainly by MITRE, a U.S. non-profit organization.
- **Cyberattack**
An attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset. [ISO/IEC 27000:2018]
- **Cybersecurity**
To prevent the leak or falsification of electronic data as well as the malfunction of IT or control systems against expected behavior.
- **ISMS (Information Security Management System)**
A framework to operate a system by determining the required security level, establishing a plan and distributing resources through its own risk assessment in order to manage an organization. The requirements are defined in the international standard ISO/IEC 27001.
- **Malware**
Software or firmware intended to perform an unauthorized process that will have adverse impact on the confidentiality, integrity, or availability of an information system. A virus, worm, Trojan horse, or other code-based entity that infects a host. Spyware and some forms of adware are also examples of

malicious code. [NIST SP 800-53 Rev.4]

- OWASP (Open Web Application Security Project)
An open source software community that aims to share information and raise awareness about software security, including the Web.
- Protocol
Predetermined mass of rules and steps for parties, so that more than one party can smoothly transmit signals, data, and information to one another.
- PSIRT (Product Security Incident Response Team)
An organization that responsible for improving the security of the company's products and responding to incidents when they occur.
- Risk
The effect of uncertainty on objectives. [ISO/IEC 27000:2018]
- Supply Chain
A linked set of resources and processes between multiple tiers of developers that begins with the sourcing of products and services and extends through the design, development, manufacturing, processing, handling, and delivery of products and services to the acquirer. [ISO 28001:2007, NIST SP 800-53 Rev.4]
- Threat
A potential cause of an undesirable incident that could damage the system or the organization.
- Threat Analysis
Identifying threats to devices, software, systems, etc., and evaluating their impact. Threat analysis is mainly done in the product requirements definition and design phase.
- Threat Intelligence
Information that may be useful in protecting against threats, detecting attacker activity, responding to threats, etc. [NIST SP 800-150]
- Vulnerability
A weakness of an asset or control (3.14) that can be exploited by one or more threats. [ISO/IEC 27000:2018]

10.3. Reference information

10.3.1. Reference documents for SBOM

This section provides a list of reference documents on SBOM published by domestic and foreign government agencies.

- **U.S. NTIA : Roles and Benefits for SBOM Across the Supply Chain (November 2019)**

A document summarizing the benefits of using SBOM from the perspective of software developers, purchasers, and users. Benefits are described by cost, security, licensing, compliance., and software stability in the supply chain.

https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf

- **U.S. NTIA : Software Bill of Materials (SBOM) (August 2020)**

A document summarizing the background of the study of SBOM and the role and effectiveness of SBOM in the software ecosystem and providing an overview of SBOM.

https://www.ntia.gov/files/ntia/publications/sbom_overview_20200818.pdf

- **U.S. NTIA : SBOM FAQ (November 2020)**

A collection of FAQs on SBOM overview, utilization effects, SBOM creation and distribution.

https://www.ntia.gov/files/ntia/publications/sbom_faq_-_20201116.pdf

- **U.S. NTIA : Sharing and Exchanging SBOMs (February 2021)**

A document describing options for how SBOM data can be shared along the supply chain, with the goal of minimizing the burden on the suppliers who created SBOM data and on the users of the SBOM

https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf

- **U.S. NTIA : SBOM Tool Classification Taxonomy (March 2021)**

A document showing the classification of SBOM tools. It classifies the purpose of use of tools into three categories: producing, consuming, and transferring SBOMs, and organizes the types of tools for each purpose.

https://www.ntia.gov/files/ntia/publications/ntia_sbom_tooling_taxonomy-2021mar30.pdf

- **U.S. NTIA : Software Identification Challenges and Guidance (March 2021)**

A document describing the challenges of uniquely identifying software

components internationally. The purpose of the document is to provide strategies and guidance for addressing the challenges.

https://www.ntia.gov/files/ntia/publications/ntia_sbom_software_identity-2021mar30.pdf

- **U.S. NTIA : SBOM at a Glance (April 2021)**

A document summarizing how to use SBOM and the role of SBOM in ensuring transparency of the software supply chain while listing reference documents. The document also includes information that should be included in SBOM. The document is also translated into Japanese by JPCERT/CC.

https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_apr2021.pdf

https://www.ntia.gov/files/ntia/publications/sbom_at_a_glance_ja.pdf

- **U.S. NTIA : SBOM Options and Decision Points (April 2021)**

A document intended to help clarify what is feasible with the current method with respect to SBOM and the needs of suppliers and users of SBOM.

https://www.ntia.gov/files/ntia/publications/sbom_options_and_decision_points_20210427-1.pdf

- **U.S. NTIA : The Minimum Elements For a Software Bill of Materials (SBOM) (July 2021)**

A document that defines the minimum elements of the SBOM. The minimum elements are divided into three categories, and the outline of each category and specific items to be included in the SBOM are defined.

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

- **U.S. NTIA : Vulnerability-Exploitability eXchange (VEX) – An Overview (September 2021)**

A document that provides an overview of VEX, which is an indicator to judge whether a particular software component is affected by a vulnerability or not. VEX represents the status of vulnerability in a particular product. The document expresses the status in four levels.

https://www.ntia.gov/files/ntia/publications/vex_one-page_summary.pdf

- **U.S. NTIA : How-To Guidance for SBOM Generation (October 2021)**

A document summarizing two points of view as a Guidance for SBOM generation: how to collect information for collection method for SBOM generation and how to generate a specific SBOM. Although this Guidance was developed through the SBOM PoC in the healthcare field by NTIA, it is expected to be used not only in the healthcare field but also in the generation

of SBOM in all industries.

https://www.ntia.gov/files/ntia/publications/howto_Guidance_for_sbom_generation_v1.pdf

- **U.S. NTIA : Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) (Initial version: November 2019, Revised: October 2021)**

A document that presents the concept of SBOM, related terminology, and basic ideas about the representation of software components, as well as the process of creating SBOM.

https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf

- **U.S. NTIA : SBOM Myths vs. Facts (November 2021)**

A document that organizes typical myths about SBOM and facts to solve them, with the aim of correctly showing the benefits of SBOM.

https://www.ntia.gov/files/ntia/publications/sbom_myths_vs_facts_nov2021.pdf

- **U.S. NTIA : Software Suppliers Playbook: SBOM Production and Provision (November 2021)**

A playbook on SBOM generation for software suppliers. This playbook covers three topics: “procedures for SBOM production”, “considerations for SBOM production”, and “supplementary information about SBOM”.

https://www.ntia.gov/files/ntia/publications/software_suppliers_sbom_production_and_provision_-_final.pdf

- **U.S. NTIA : Software Consumers Playbook: SBOM Acquisition, Management, and Use (November 2021)**

A playbook for software users on the use of SBOM. This playbook summarizes the points to be considered when acquiring SBOM from suppliers, the process and platform for utilizing SBOM, and the intellectual property and confidentiality of SBOM.

https://www.ntia.gov/files/ntia/publications/software_consumers_sbom_acquisition_management_and_use_-_final.pdf

- **U.S. NTIA : Survey of Existing SBOM Formats and Standards - Version 2021 (Initial version : 2019, Revised : 2021)**

A document that summarizes the results of a survey on existing SBOM formats and standards, in addition to future issues. As for the existing SBOM formats, SPDX, CycloneDX, and SWID are outlined, with use cases and

features.

https://www.ntia.gov/files/ntia/publications/sbom_formats_survey-version-2021.pdf

- **U.S. NIST : SP 800-218 Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities (February 2022)**

A framework document that summarizes methodologies for software developers to mitigate software vulnerabilities. The methodologies are classified into four categories, and tasks for practicing each methodology are systematically organized.

<https://csrc.nist.gov/publications/detail/sp/800-218/final>

- **U.S. CISA : Vulnerability Exploitability eXchange (VEX) – Use Cases (April 2022)**

A document showing the minimum elements to be included in a VEX document. In addition, use cases are presented as concrete examples for creating VEX documents.

https://www.cisa.gov/sites/default/files/publications/VEX_Use_Cases_Document_508c.pdf

- **U.S. CISA : Vulnerability Exploitability eXchange (VEX) - Status Justifications (June 2022)**

A document that defines five specific arguments to justify the “NOT AFFECTED” status among the “Vulnerability Status” in the minimum elements of the VEX document.

https://www.cisa.gov/sites/default/files/publications/VEX_Status_Justification_Jun22.pdf

- **U.S. CISA, NSA, ODNI : Securing Software Supply Chain Series - Recommended Practices for Developers (September 2022)**

A document that provides recommendations for software developers to ensure a secure software supply chain. This document is the first part of a three-part guidance series focusing on the roles of software developers, software suppliers, and software users. The document recommends the creation of SBOM for software containing third-party components, vulnerability assessments.

https://www.cisa.gov/uscert/sites/default/files/publications/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF

- **U.S. CISA, NSA, ODNI : Securing Software Supply Chain Series -**

Recommended Practices for Suppliers (October 2022)

A document that provides recommendations for software suppliers to ensure a secure software supply chain. This document is the second part of a three-part guidance series focusing on the roles of software developers, software suppliers, and software users. The document recommends that suppliers act as an intermediary between developers and users to protect software and to respond to and notify users of vulnerabilities.

https://media.defense.gov/2022/Oct/31/2003105368/-1/-1/0/SECURING_THE_SOFTWARE_SUPPLY_CHAIN_SUPPLIERS.PDF

- **U.S. CISA, NSA, ODNI : Securing Software Supply Chain Series - Recommended Practices for Customers (November 2022)**

A document that provides recommendations for software users to ensure a secure software supply chain. This document is the third part of a three-part guidance series focusing on each of the three roles of software developers, software suppliers, and users. The document recommends requesting SBOM from suppliers and evaluating software vulnerabilities based on SBOM.

https://media.defense.gov/2022/Nov/17/2003116445/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_CUSTOMER.PDF

- **U.S. CISA : Software Bill of Materials (SBOM) Sharing Lifecycle Report (April 2023)**

A report on the SBOM sharing lifecycle. It identifies three basic phases before SBOM is shared from the creator to the users, with an overview of each phase and the degree of sophistication for each phase. The degree of sophistication represents the relative amount of cost and resources required to implement each phase, and is defined as low, medium, or high. In addition, in order to help understand the current status of SBOM sharing, the report presents the results of interviews with concerned organizations on how their organizations are sharing SBOM.

<https://www.cisa.gov/resources-tools/resources/software-bill-materials-sbom-sharing-lifecycle-report>

- **U.S. CISA : Minimum Requirements for Vulnerability Exploitability eXchange (VEX) (April 2023)**

A document that describes the minimum requirements for a VEX document. The document presents the items that constitute a VEX document and the elements included in each item and defines the essential items and essential requirements for each of them. The document regards the mandatory

requirements as the minimum requirements of VEX documents.

<https://www.cisa.gov/resources-tools/resources/minimum-requirements-vulnerability-exploitability-exchange-vex>

- **U.S. CISA : Types of Software Bill of Materials (SBOM) (April 2023)**

A document defining the types of SBOM. This document categorizes the types of SBOM that may be generated in each phase of the software lifecycle and presents general SBOM generation methods, advantages, and limitations of each type.

<https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom>

- **Netherland NCSC : SBOM startersgids (July 2023)**

A Guidance to support the introduction of SBOM in organizations. This document outlines the basic knowledge of SBOM and VEX, as well as the processes for organizations to create, manage, and share SBOM, and tips for working with suppliers. In addition, it explains the typical vulnerability identifiers and shows how to use SBOM in vulnerability management within organizations.

<https://www.ncsc.nl/documenten/publicaties/2023/juli/5/sbom-startersgids>

- **Germany BSI : Technische Richtlinie TR-03183: Cyber-Resilienz-Anforderungen an Hersteller und Produkte (First edition: August 2023; Revised: January 2024)**

Technical Guidelines that set out the requirements for SBOM. These Guidelines are mainly aimed at software vendors, and set out the requirements for SBOM formats and technical requirements.

https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03183/TR-03183_node.html

- **U.S. CISA : Software Identification Ecosystem Option Analysis (October 2023)**

A white paper that outlines the main requirements for achieving an ecosystem for software identification and the specific methods for achieving them. This document outlines the requirements for identifier availability and granularity, as well as the methods for achieving each requirement.

<https://www.cisa.gov/sites/default/files/2023-10/Software-Identification-Ecosystem-Option-Analysis-508c.pdf>

- **U.S. CISA 、 NSA 、 ODNI : Securing the Software Supply Chain: Recommended Practices for Software Bill of Materials Consumption**

(November 2023)

Guidance on the use of SBOM to ensure security in the software supply chain. This guidance provides principles and best practices for the use of SBOM by software users (e.g., suppliers, developers, organizations that acquire OSS and third-party software).

<https://media.defense.gov/2023/Nov/09/2003338086/-1/-1/0/SECURING%20THE%20SOFTWARE%20SUPPLY%20CHAIN%20RECOMMENDED%20PRACTICES%20FOR%20SOFTWARE%20BILL%20OF%20MATERIALS%20CONSUMPTION.PDF>

- **U.S. CISA : When to Issue VEX Information (November 2023)**

This document provides examples of the organization and function that issues VEX information (Who), and the timing at which VEX information is issued (When). The document also provides information on VEX considerations in the software supply chain.

<https://www.cisa.gov/sites/default/files/2023-11/When-to-Issue-a-VEX-508c.pdf>

- **U.S. CISA, NSA, ODNI : Securing the Software Supply Chain: Recommended Practices for Managing Open-Source Software and Software Bill of Materials (December 2023)**

A document that sets out recommended practices for managing OSS and SBOM to ensure a secure software supply chain. This document sets out recommended practices for seven themes related to the management of OSS and SBOM.

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

- **U.S. NSA : Recommendations for Software Bill of Materials (SBOM) Management (January 2024)**

A document that emphasizes best practices and provides recommendations so that users of the National Security System (NSS) can incorporate SBOM management functions that meet their needs for managing cybersecurity supply chain risks. It includes recommendations for software suppliers and users, specific SBOM management Guidelines for the NSS, and recommendations for the functions of SBOM management tools.

<https://media.defense.gov/2023/Dec/14/2003359097/-1/-1/0/CSI-SCRM->

[SBOM-MANAGEMENT.PDF](#)

- **U.S. CISA : SBOM Sharing Roles and Considerations (March 2024)**
A document that describes the three roles in the SBOM shared lifecycle (SBOM author, SBOM distributor, SBOM consumer) and the factors to be considered when working on the three phases of the shared lifecycle (Discovery, Access, Transport).
<https://www.cisa.gov/resources-tools/resources/sbom-sharing-roles-and-considerations>
- **U.S. CISA : SBOM Sharing Primer (May 2024)**
This document presents several examples of how to share SBOM in the software supply chain. Each example is classified into three levels of sophistication (Low, Medium, and High), and companies can use these examples to determine the level of sophistication of their own SBOM sharing processes.
<https://www.cisa.gov/resources-tools/resources/sbom-sharing-primer>

10.3.2. SBOM Tools

This section shows some examples of SBOM tools that contribute to the creation, operation, and management of SBOM. Not only commercial SBOM tools but also OSS SBOM tools are available, and each tool has its own characteristics. Organizations implementing an SBOM should select appropriate SBOM tools based on the purpose of SBOM introduction and the scope of SBOM application. The tools listed in this section are only examples available for reference at the time of preparation of this Guidance. It is to be noted that the use of any tool is not recommended. For appropriate tool selection, it is desirable to evaluate and select various tools existing in the market, not limited to the tools described in this section, based on the viewpoints described in Section 4.2.

(1) Commercial tools

*In alphabetical order

No.	Name	Developer	Features
1	Black Duck	Synopsys, Inc.	<ul style="list-style-type: none">• Multiple scanning approaches, including code matching, container analysis, and binary analysis, are available for accurate and efficient analysis.• For vulnerability management, it enables rapid vulnerability detection by leveraging vulnerability information from NVD and proprietary sources.• It quantifies and manages risk in terms of security, licensing, compliance, operations, etc.• It provides a Japanese-language GUI.
2	Checkmarx SCA	Checkmarx Ltd.	<ul style="list-style-type: none">• Hosting many repositories on GitHub allows automatic tracking of OSS in use.• For vulnerability management, it detects vulnerable OSS packages in the source code and provides remedies.• It visualizes OSS license risks and enables effective license management.
3	Cybellum	Cybellum Technologies Ltd.	<ul style="list-style-type: none">• Regarding vulnerability management, it automatically detects vulnerabilities in the target software products and provides the priority order for addressing the detected vulnerabilities and mitigation measures.• It continuously monitors software products and can detect vulnerabilities in software update programs and new versions of components.• By importing and centrally managing multiple SBOM, it is possible to integrate the SBOM operation process within an organization.

No.	Name	Developer	Features
4	Finite State Platform	Finite State, Inc.	<ul style="list-style-type: none"> • It provides an integrated management platform for SBOM and vulnerabilities. • It supports the architecture of various embedded devices and is able to generate SBOM by visualizing components from binaries and firmware. • It allows for centralized management of vulnerabilities by importing the results of diagnostics from over 150 other companies' tools. • It allows for triage of vulnerabilities with high urgency, such as attack occurrence status, and determination of response priority. • It has multiple delivery services, such as SaaS, private cloud, and on-premises.
5	FOSSA	FOSSA, Inc.	<ul style="list-style-type: none"> • It detects vulnerabilities and continuously monitors risk while providing necessary solutions for effective vulnerability management. • It facilitates compliance. and license management with high-quality policy features, powerful scanning, and flexible reporting. • It automates and streamlines SBOM management in Agile and DevOps processes through integration with development environments. • It has multiple report formats, including SPDX, and the ability to import multiple SBOM formats for vulnerability management.

No.	Name	Developer	Features
6	FossID	FossID AB	<ul style="list-style-type: none"> • It detects not only components, packages, and libraries, but also snippets of OSS. • It detects vulnerable software by analysis based on snippet-level information, rather than by component- and version-based analysis. • It generates and manages SBOM in SPDX format, including license, copyright, vulnerability, etc. information. • It visualizes the risk of license violations for a wide range of OSS, including strong/weak copyleft and non-commercial licenses with respect to license management.
7	Insignary Clarity	Insignary, Inc.	<ul style="list-style-type: none"> • It analyzes binary files to identify encompassing components (no source code or reverse engineering required). • It analyzes binary files using patterns, making it independent of the build environment. • It is applicable to cloud and on-premises software. • It can be easily deployed due to the cloud-type solution.
8	MEND SCA	WhiteSource Software, Inc.	<ul style="list-style-type: none"> • It detects OSS libraries and frameworks used in cloud services, desktop applications, embedded software, etc. without false negatives. • In vulnerability management, it issues an alert immediately when a vulnerability occurs, using its own vulnerability database, which is always kept up to date. Also, it provides impact and severity scores and detailed information about how to resolve them. • In license management, it integrates this tool into the development environment of the target software, such as IDEs and package managers, to enable developers to automatically identify OSS license information each time they add a new OSS component.

No.	Name	Developer	Features
9	MergeBase	MergeBase Software Inc.	<ul style="list-style-type: none"> • It integrated software supply chain security solution for creating and managing SBOM. • It scans codes and binaries to create SBOM that include vulnerability information, license information, age information, and transitive dependency information. • It reduces the effort required to fix code by reducing vulnerability-related risks and proposing improvement methods. • It integrates with JVN to enable vulnerability analysis and management. • It partners with domestic partner companies to provide Japanese-language documentation and technical support in Japanese.
10	Revenera SCA	Flexera Software LLC	<ul style="list-style-type: none"> • It enables the creation of accurate SBOM, by not only analyzing source code, binaries, and other software analysis, but also by collating proprietary OSS knowledge databases and third-party SBOM data. • It enables effective vulnerability management by utilizing multiple sources such as NVD and our own vulnerability database (Secunia Research).
11	Snyk	Snyk, Ltd.	<ul style="list-style-type: none"> • It can be integrated into existing IDEs, repositories, and workflows. • It uses advanced security intelligence to monitor vulnerabilities during targeted software development. • It provides practical remediation advice on vulnerabilities and other issues related to vulnerability management.

No.	Name	Developer	Features
12	Sonatype Lifecycle	Sonatype, Inc.	<ul style="list-style-type: none"> Available by integrating the tool into the development environment of the target software, such as an IDE or source code control system. In vulnerability management, it issues alerts quickly by continuously monitoring for vulnerabilities in software and components and the risk level of vulnerabilities.
13	Veracode SCA	Veracode, Inc.	<ul style="list-style-type: none"> It enables the creation of an SBOM in CycloneDX format as a list of OSS components. In vulnerability management, it provides information about vulnerabilities detected and how to address them, as well as prioritization of vulnerabilities to be addressed. In license management, it detects OSS license violation risks and manages license compliance.
14	yamory	Assured, Inc.	<ul style="list-style-type: none"> It detects and manages software vulnerabilities used in the target IT systems. In vulnerability management, it automatically determines the priority of vulnerabilities with the auto-triage function. In addition, updates the vulnerability database daily, enabling early detection of urgent vulnerabilities. In license management, it visualizes the risk of OSS license violations.

(2) OSS tools

*In alphabetical order

No.	Name	Developer	Features
1	Augur	CHAOSS	<ul style="list-style-type: none"> It collects data on software repositories and normalizes them into a data model. It collects data on OSS projects from many sources.

No.	Name	Developer	Features
2	BOM Doctor	Sonatype, Inc.	<ul style="list-style-type: none"> • It generates SBOM by specifying a project URL or package URL on GitHub. • It visualizes generated SBOM on a tree including dependencies of components (it is also possible to visualize SBOM by uploading an existing SBOM in CycloneDX format). • It performs scoring of target software by evaluating whether it uses non-fragile components, violates licenses, etc.
3	Checkov	Bridgecrew, Inc.	<ul style="list-style-type: none"> • It is a static code analysis tool for IaC and can be used also as an SBOM tool for images and OSS packages. • The scan results can be displayed in CLI, CycloneDX, JSON, JUnit XML, CSV, SARIF, and Markdown formats.
4	Daggerboard	NewYork-Presbyterian Hospital	<ul style="list-style-type: none"> • It provides a dashboard to view and manage SBOM and related vulnerabilities immediately and can import SPDX or CycloneDX files for vulnerability detection.
5	Dependency-Track	OWASP Foundation	<ul style="list-style-type: none"> • It can identify and manage known vulnerabilities in third-party and open-source components, by leveraging multiple sources such as NVD, GitHub Advisories, etc. • It allows the identification of license information for software components. • API-first design allows easy integration with other systems.
6	FOSSology	Linux Foundation	<ul style="list-style-type: none"> • It cannot identify the name and version of the OSS, but it can detect and managing the licenses and copyrights of the components included in the target software. • It allows import and analysis using a Web UI.

No.	Name	Developer	Features
7	in-toto	Linux Foundation	<ul style="list-style-type: none"> It provides a framework for protecting the integrity of the software supply chain by ensuring that software has not been tampered with during distribution within the supply chain.
8	mjcheck4	Information-technology Promotion Agency, Japan (IPA)	<ul style="list-style-type: none"> By utilizing our own vulnerability database, it provides information on vulnerabilities and vulnerability countermeasures contained in software products. It supports SBOM import/export.
9	OSS Review Toolkit (ORT)	Linux Foundation	<ul style="list-style-type: none"> It allows SBOM creation without the need to modify existing project source code, such as applying build system plug-ins. It allows evaluation of software licenses in use, based on customizable policy rules and license classifications.
10	OSV-Scanner	Google	<ul style="list-style-type: none"> It can import SBOM written in CycloneDX or SPDX format. By utilizing own vulnerability database, it can provide vulnerability information for each component of the SBOM.
11	SBOM Tool	Microsoft Corporation	<ul style="list-style-type: none"> It integrates with various package management systems such as NPM, NuGet, PyPI, etc. to automatically detect and create SBOM in SPDX format. It runs on Windows, Linux, and macOS platforms.
12	ScanCode.io	nexB, Inc.	<ul style="list-style-type: none"> It scripts and automates the process of Software Configuration Analysis (SCA). It identifies OSS components and their license information in an application's code base.

No.	Name	Developer	Features
13	Scancode Toolkit	nexB, Inc.	<ul style="list-style-type: none"> • A standalone command line tool, easy to install, run, and integrate into the CI/CD processing pipeline. • It allows the saving of scan results in JSON, HTML, CSV, SPDX, and proprietary formats. • In license management, it allows users to identify and manage license information for OSS components by using their own extensible discovery rules.
14	SW360	Eclipse Foundation	<ul style="list-style-type: none"> • It identifies and manages security vulnerability information for software components. • It identifies and manages license information for software components.
15	SwiftBOM	CERT Coordination Center (CERT/CC)	<ul style="list-style-type: none"> • It allows manual input for SBOM generation. • It imports previously created SBOM and displays SBOM in a tree-like view.
16	Syft & Grype	Anchore Enterprise	<ul style="list-style-type: none"> • It seamlessly integrates SBOM generation by Syft and vulnerability detection by Grype. • It converts SBOM information between SBOM formats such as CycloneDX, SPDX, and Syft's own format. • It detects and manages major vulnerabilities in OS packages and language packages.
17	Trivy	Aqua Security Software, Ltd.	<ul style="list-style-type: none"> • It detects and manages various security issues such as known vulnerabilities, IaC misconfigurations, etc. • It scans various targets such as container images, file systems, etc.

10.3.3. SBOM data formats

The SBOM “Minimum Elements” include a category of “Automation Support”, which considers support for automation in the automatic generation, readability of SBOM, etc. As specific data formats, three formats—SPDX, CycloneDX, and Software Identification Tags (SWID tags)—have been discussed internationally. In addition to these three formats, the following section outlines SPDX Lite, a format developed by Japan based on SPDX. The SBOM data format is a standard for exchanging SBOM across organizations. The selection of the data format and the data fields to be included in an SBOM should be decided upon agreement between the SBOM user and the supplier.

(1) SPDX

SPDX was developed by a project under the Linux Foundation and recognized as an international standard for the SBOM format in September 2021 as an ISO/IEC 5962:2021 standard. The detailed specification of SPDX is available on the website,⁵⁵ and the project continues to study and update it. In April 2024, a new version, SPDX v3.0, was announced. In SPDX v3.0, the focus is on security licenses, AI, datasets, and software construction processes to accommodate more common SBOM generation and usage use cases.

In the following, as an overview of the SPDX v2.3.0, the format structure, examples, and purposes of use of the format, and features of the format are described.

1) Format configuration

SBOM in the SPDX format contain information about components created according to the SPDX Specification, license, and copyright. Tag:Value(txt), RDF⁵⁶, XLS, JSON⁵⁷, YAML⁵⁸, and XML⁵⁹ formats are supported. Sections and items

⁵⁵ <https://spdx.GitHub.io/spdx-spec/v2.3/>

⁵⁶ As a method of analyzing RDF format files, it is known, for example, to utilize the SPARQL language to search and manipulate data described in the file.

⁵⁷ As a method of analyzing a json format file, for example, it is known to utilize the jq command to obtain necessary information from the file.

⁵⁸ By using tools that support YAML format files, such as Visual Studio Code and IntelliJ IDEA, it is easy to view and analyze files.

⁵⁹ As a method of analyzing xml format files, for example, it is known to utilize the xmllint command to obtain the necessary information from the file.

classified into each section are specified as contents to be included in an SBOM document. A summary of each section is given below. Only the section “Creation Information” is defined as mandatory. Other sections that are not mandatory are used when the SBOM document author judges that they should be included in the SBOM. In addition, the items defined in each section that must be included if the relevant section is used are also defined.

- **Creation Information [Mandatory section]:**

A section where the supplier provides the SBOM document and presents the information (e.g., SPDX version, SBOM data license, and author) necessary for the user to use the SBOM document. This section needs to be included in every SBOM document with SPDX.

- **Package Information:**

A section in the SBOM that presents information necessary to group products, containers, components, etc.

- **File Information:**

A section that presents information (name, checksum, license, copyright, etc.) about the files of a product, container, components, etc.

- **Snippet Information:**

A section that is used when a file is generated from another resource. This section is useful to indicate that part of a file has been copied from another file.

- **Other Licensing Information:**

SPDX defines a license list called SPDX License List to show licenses for file information. In the “Package Information”, “File Information”, and “Snippet Information” sections, the license information for the package, file, or snippet to be described is selected from the SPDX license list. However, the SPDX License List does not cover all licenses for packages, files, and snippets. Therefore, it is possible to present license information other than the SPDX License List (such as restrictions by proprietary software) in this section.

- **Relationships:**

A section that presents the relationships between files and packages such as products, containers, and components in the SBOM.

- **Annotations:**

A section that is used to review the SBOM and share the information obtained from the review with others. In addition, this section can be used by SBOM

document authors who wish to store information in an SBOM that does not apply to the other sections or items mentioned above.

2) Examples and purposes of use

The following examples and purposes of use are expected regarding the SPDX:

- Describing relationships between system components,
- Managing intellectual property (licenses, copyrights) of software components,
- Performing a risk assessment of the software supply chain and validating components,
- Creating an inventory of software components, container content, etc.,
- Tracking executables back to individual source files and source snippets,
- Identifying lines of code embedded in files, and
- Associating CPE, SWHID (SoftWare Heritage persistent IDentifiers), and package URLs, which are formats for uniquely identifying software, with specific packages to facilitate additional security analysis.

3) Data format features

The SPDX has the following features:

- Ability to extend beyond snippets and files to include packages, containers, and OS distributions, as software for which SBOM are created,
- Ability to verify whether SBOM data has been tampered with in deliverables created as SBOM documents, by using the provided hash value,
- Having an extensive list of intellectual property and license information (SPDX license),
- Ability to integrate with other package reference systems and security systems, and
- Ability to logically partition documents related to complex systems and manage them in sections or items of the SBOM document.

(2) SPDX Lite

SPDX Lite is a format developed by the OpenChain Japan Work Group (WG) license information subgroup, which is mainly active for Japanese companies in the OpenChain Project, a project under the umbrella of the Linux Foundation. SPDX Lite is included in part of the ISO/IEC 5962:2021 standard for SPDX and is defined as being included in SPDX. The detailed specifications of SPDX Lite are published on the website⁶⁰ as part of the SPDX v2.3.0 specifications. The following presents, as an overview of the SPDX Lite, the structure of the format and specific items, usage examples and purposes of the format, and the characteristics of the format.

1) Format configuration and specific items

An SBOM in SPDX Lite format contains information such as components, licenses, and copyrights, and supports Tag-Value (txt), RDF, XLS, JSON, YAML, and XML formats. The content to be included in an SBOM document consists of the mandatory items and other basic information classified into the “Creation Information” and “Package Information” sections in SPDX described above. The items required for SPDX Lite are as follows:

Table 10-2 Relationship between SPDX Lite items and SPDX

Section name in SPDX	Item name in SPDX-Lite
Creation Information	SPDX Version
	Data License
	SPDX Identifier
	Document Name
	SPDX Document Namespace
	Author
	Created
Package Information	Package Name
	Package SPDX Identifier
	Package Version
	Package File Name
	Package Supplier
	Package Download Location
	Files Analyzed

⁶⁰ <https://spdx.GitHub.io/spdx-spec/v2.3/SPDX-Lite/>

Section name in SPDX	Item name in SPDX-Lit
	Package Home Page
	Concluded License
	Declared License
	Comments on License
	Copyright Text
	Package Comment
	External Reference field
Other Licensing Information	License Identifier
	Extracted Text
	License Name
	License Comment

2) Examples and purposes of use

The following examples and purposes of use are expected regarding the SPDX Lite:

- Manually managing only, the mandatory fields that are classified in the SPDX section of the “Creation Information” and “Package Information” and
- Creating SBOM that are not at the level of SPDX but rather correspond to the minimum required fields in the automotive industry and consumer electronics industry with an emphasis on usability.

3) Data format features

SPDX Lite has the following features:

- Ability to manage SBOM with a focus on operability, as it contains only the minimum required items compared to SPDX,
- High SBOM tool compatibility with SPDX, as it contains mandatory fields that fall under the “Document Information” and “Package Information” sections of SPDX, and
- Ability to manually create SBOM documents in SPDX Lite format without the need for specialized tools.

(3) CycloneDX

CycloneDX was developed by a project of the OWASP community with the goal of

developing a fully automated, security specific SBOM format standard. The detailed specifications of the CycloneDX are available on the web site⁶¹ and are being maintained and updated by the core working group of the OWASP community. A new version, CycloneDX 1.6, was released in April 2024.

As an overview of the CycloneDX v1.5, the following provides the structure of the format, examples of use and purpose of the format, and features of the format.

1) Format configuration

SBOM in the CycloneDX format contain information about components, and licenses, copyrights. The JSON, XML, and Protocol Buffers (protobuf) formats are supported. An SBOM document must include object models and fields that are classified into each object model. An overview of each object model is shown below. In addition, the items specified in each object model that must be included when the relevant model is used are also defined. Although not classified as an object model, the SBOM document must be in the CycloneDX format and must include an item for the CycloneDX version and the SBOM document version.

- **SBOM Metadata :**

An object model that presents information about the supplier, the developer, the scope of the software covered by the SBOM document, the tools used to create the SBOM document, etc.

- **Components :**

An object model that presents an inventory of first and third-party software components. This object model can include information about software components, including type, ID, license, copyright, cryptographic hash function, provenance, history, and changes made. In addition, this object model can represent a combination of components, and a combined component can have various information as a single component. Furthermore, it is possible to apply a digital signature to components and combined components.

- **Services:**

An object model that presents information about external APIs that may be invoked by the software covered by the SBOM document. This object model can include information such as the endpoint URI of the external API, authentication requirements, trust boundaries with the external API, and

⁶¹ <https://cyclonedx.org/docs/1.4/json/>

data flow and classification between services. Furthermore, it is possible to apply digital signatures to services.

- **Dependencies:**

An object model that presents dependencies between components and other components. It can represent not only components among components but also components that depend on services and services that depend on services. Dependencies can also represent transitive dependencies.

- **Compositions:**

An object model that presents each component (including components, services, and dependencies) and the completeness of the component within the SBOM. The aggregate of each composition can be described as "complete", "incomplete", "incomplete first party only", "incomplete third-party only", or "unknown". With this object model, it is possible to understand how complete the created SBOM is and whether there are components in the SBOM where completeness is unknown.

- **Vulnerabilities:**

An object model that presents known vulnerabilities and their exploitability in third-party software and OSS is included in the SBOM. It can also present unknown vulnerabilities affecting components and services and can be used as a security advisory for VEX, etc.

- **Extensions :**

An object model that enables experimentation of new functions in CycloneDX and support for specialized and future use cases. the CycloneDX project encourages community participation and development targeting extensions for specialized and industry-specific use cases.

2) Examples and purposes of use

The following examples and purposes of use are expected regarding the CycloneDX:

- Describing the components of a system and the relationships between components,
- Managing intellectual property (licenses, copyrights) for software components,
- Performing a risk assessment of the software supply chain and validating

components,

- Creating an inventory of software components, container content, etc.,
- Tracking executables back to source files and source snippets,
- Identifying the source of code embedded in files,
- Associating formats for uniquely identifying software (such as CPE, SWID, package URL) with specific packages, thus facilitating additional security analysis,
- Validating the integrity of signed or combined components and the SBOM, and
- Using as a convenient format for creating and distributing software when building software and as a binary format for M2M (machine-to-machine).

3) Data format features

The CycloneDX has the following features:

- An SBOM format with security management in mind, allowing the imputing of information about known vulnerabilities and their exploitability,
- A security related SBOM format for various types of software, including applications, components, services, firmware, and devices, used in a wide range of industries and suitable for commercial use,
- A format consisting of a structured object model, which enables one to easily learn and implement,
- Achieving automation when integrated with many development ecosystems, and
- Extensible specifications allow a rapid trial of new functions to meet organizational and industry-specific requirements.

(4) SWID Tag

Software Identification (SWID) Tags were designed to provide a transparent way for organizations to track the software installed on their managed devices. It was defined by ISO in 2012 and updated as ISO/IEC 19770-2:201523 in 2015. As part of the software installation process along the software lifecycle, when software is installed on a device, information about the installed software called a tag, is

attached to the device, and when the software is uninstalled, the tag is removed. The following provides an overview of SWID tags, including the format configuration, examples, and purposes of use of the format, and the format features.

1) Format configuration

An SBOM in the SWID tag format describes information such as software installed in the device created according to the SWID tag and patches applied to the software and supports the XML format. A SWID tag defines a tag that indicates information about software installed on a device to understand the life cycle of the target device. An overview of each tag is shown below. Each tag can present information such as the tag creator, the software installed on the device, and the dependencies by linking to other software, and can be used as an SBOM of the target device.

- **Primary Tag:**
A tag that identifies and presents the software installed on the target device.
- **Patch Tag:**
A tag that identifies and presents patches that have been applied to the software installed on the target device, e.g., by updating the software.
- **Corpus Tag:**
A tag that identifies and describes software installed on the target device. This tag is used to represent software metadata such as software installation packages, installers, software updates, and patches.
- **Supplemental Tag:**
A tag that is used to add additional information to the above tags. This tag is used by device users and software management tools to add optional information.

2) Examples and purposes of use

The following examples and purposes of use are expected regarding the SWID tag:

- Creating SBOM with software installed on devices managed by the organization as a component,
- Continuously tracking software installed on devices,
- Identifying vulnerable software on endpoints,

- Ensuring whether the software installed on devices is properly patched,
- Preventing the installation of unauthorized or corrupted software,
- Preventing corrupted software from running, and
- Managing user rights and access rights for managed devices.

3) Data format features

The SWID tag has the following features:

- Updating information about each tag as it moves through the software lifecycle, so that information about software IDs created at build time can be accurately assigned to the tag and provided,
- Standardizing software information that can be exchanged between suppliers and users during software installation, and
- Enabling association of software-related information, such as relevant patches and updates, configuration settings, security policies, and vulnerability and threat advisories.