(Provisional Translation)

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

Ver. 1.0

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

July 28, 2023

Table of Contents

1. Background and objectives

1.1. Background

As industrial activities become more service-oriented, the importance of software in industry is increasing. In particular, 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 products and services that utilize software, it is essential to control software vulnerabilities. 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 also 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 in particular, to be one solution to the issue of component vulnerability management. SBOMs 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 Biden[1] in May 2021. A survey conducted by the Linux Foundation of 412 global organizations in the third quarter of 2021[2] 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 SBOMs: cost-effectiveness of SBOM introduction, issues related to sharing SBOMs in the supply chain, issues related to contracts when managing SBOMs, and issues related to the implementation of SBOMs in small and medium-sized enterprises. In light of 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 PoCs, 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

[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/

increased development productivity.

- Comparing the workloads for manual component management and the 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 tools[3].
- 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. In particular, by utilizing commercial SBOM tools, dependencies between different pieces of OSS and recursive use of OSS (reused parts) 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 a large number of 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 parts, limitations on SBOM formats that can be handled, and license false negatives.

[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.

- 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 carefully, as there may be 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 SBOMs 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.

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

1.2. Objectives

In order to solve various issues related to the creation, sharing, operation, and management of SBOMs 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 in order to support efficient and effective SBOM introduction. Although this Guidance is intended primarily for software suppliers, it can also be used and referenced by companies that procure and use software. It shall be noted that since SBOM is a method of software management, the objective is not creating SBOM, but rather the appropriate management of software using SBOM is important. 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.

This Guidance is mainly intended for those who are new to SBOMs, such as organizations that are not yet aware of the details of their efforts to implement SBOMs. 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.

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.

[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

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 7.1 of the Appendix provides a checklist of items to be implemented at each step of the SBOM introduction, which shall be referred to in conjunction with efforts to implement an SBOM.

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 Material (SBOM), a method of software management, is attracting attention in response to the threats, and the number of companies adopting it is increasing worldwide.
- SBOMs 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 SBOMs 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 SBOMs 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 Ministry of Economy, Trade and Industry (METI), vulnerability management using SBOMs reduced management workloads by about 70% compared to manual management.
- ✓ In addition, as a benefit to vulnerability management, the creation and ongoing management of SBOMs 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.
- ✓ Furthermore, as an advantage in license management, SBOMs will help reduce the risk of license violations by managing OSS license information.

[Points to using this Guidance]

- ✓ In order 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.

↑81%: Percentage of code bases containing vulnerabilities

According to a survey of 2,409 codebases published by Synopsys in 2022, the percentage of codebases containing OSS was 97%. Of those, 81% of the codebases contained at least one vulnerability[5].

↑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 year [6].

↑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 years[7].

↓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 revenue[8].

---

[5] Synopsys, 2022 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 also 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 in particular, 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 SBOMs 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.

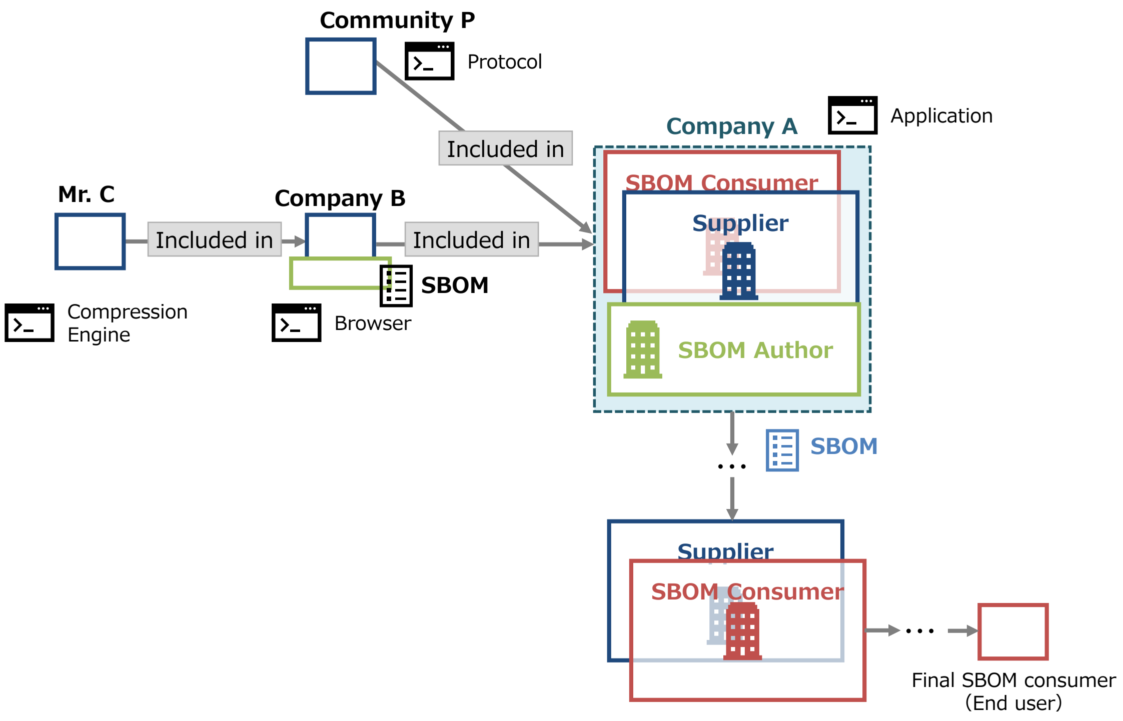


Figure 2-1 Relationship between players in the scenario

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.

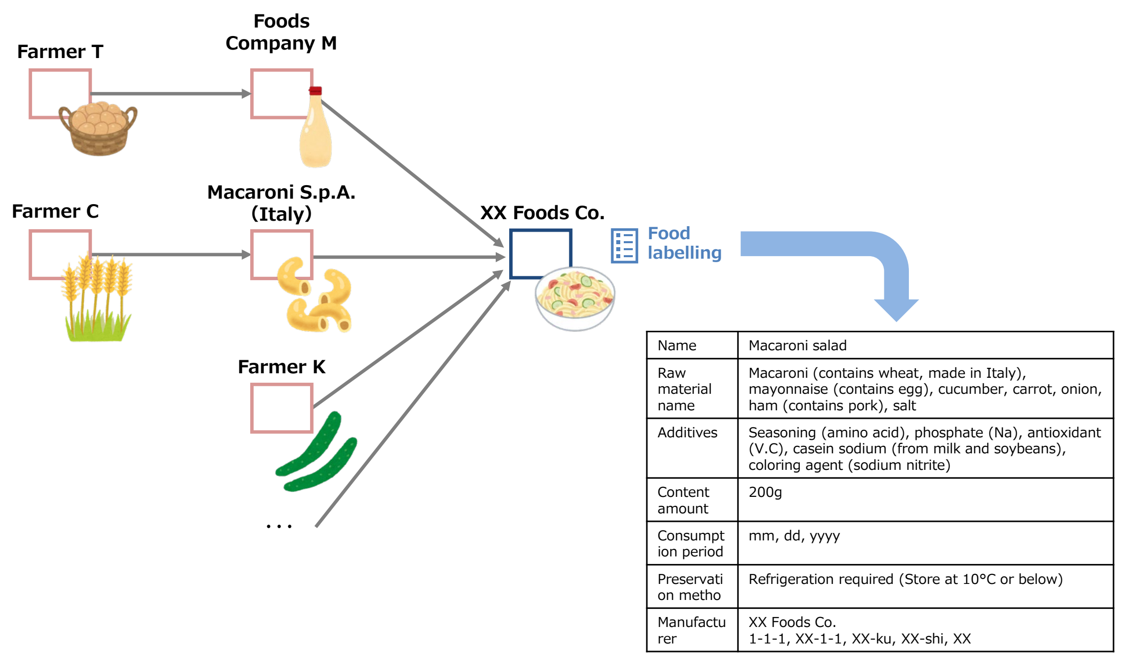
Table 2-1 Image of an SBOM in the scenario (matrix form)

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 #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	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, in order 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 similar to 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, SBOMs contribute to transparency in the software supply chain. Note that, however, SBOMs 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 SBOMs are dynamically modified even after they are created, so it is important for SBOM users to manage them.



Name	Macaroni salad
Raw material name	Macaroni (contains wheat, made in Italy), mayonnaise (contains egg), cucumber, carrot, onion, ham (contains pork), salt
Additives	Seasoning (amino acid), phosphate (Na), antioxidant (V.C), casein sodium (from milk and soybeans), coloring agent (sodium nitrite)
Content amount	200g
Consumpt ion period	mm, dd, yyyy
Preservati on metho	Refrigeration required (Store at 10°C or below)
Manufactu rer	XX Foods Co. 1-1-1, XX-1-1, XX-ku, XX-shi, XX

Figure 2-2 Conceptual image of the food supply chain and food labeling

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

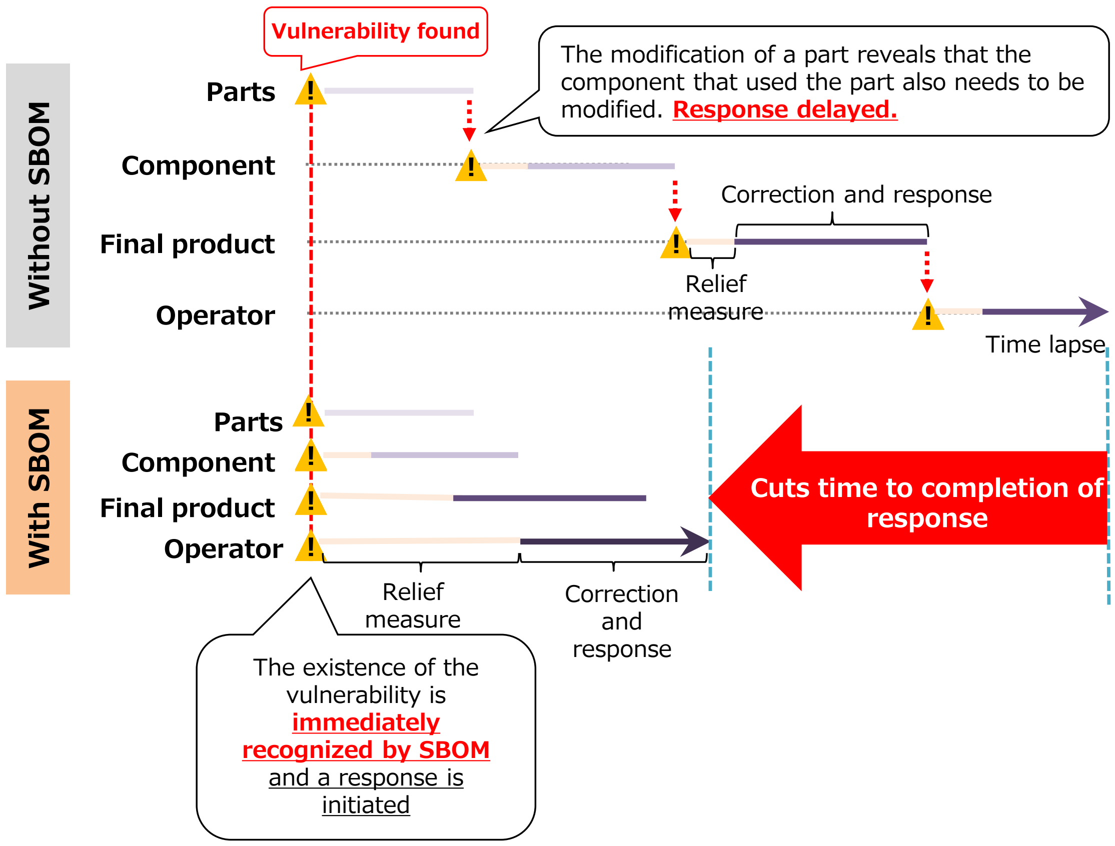
Benefit		Item	Description
Vulnerability management	Direct benefits	Reduce residual vulnerability risks	Collecting vulnerability information and matching it with SBOM information to detect vulnerabilities can reduce the risk of residual vulnerabilities in software.
		Reduce vulnerability	SBOM tools can be used to detect new vulnerabilities in

Benefit		Item	Description
		response time	real-time and determine their impact, thereby shortening 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 increases 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	Early identification of component problems reduces response costs.

Benefit		Item	Description
		costs	
		Cut development time	When selecting components to be used, workloads related to the selection are reduced by referring to past SBOMs for similar products.

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.

In order 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.



Vulnerability found
The modification of a part reveals that the component that used the part also needs to be modified. Response delayed.
Without SBOM
Parts
Component
Final product
Operator
Correction and response
Relief measure
Time lapse
With SBOM
Parts
Component
Final product
Operator
Cuts time to completion of response
Relief measure
Correction and response
The existence of the vulnerability is immediately recognized by SBOM and a response is initiated

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 or not 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.

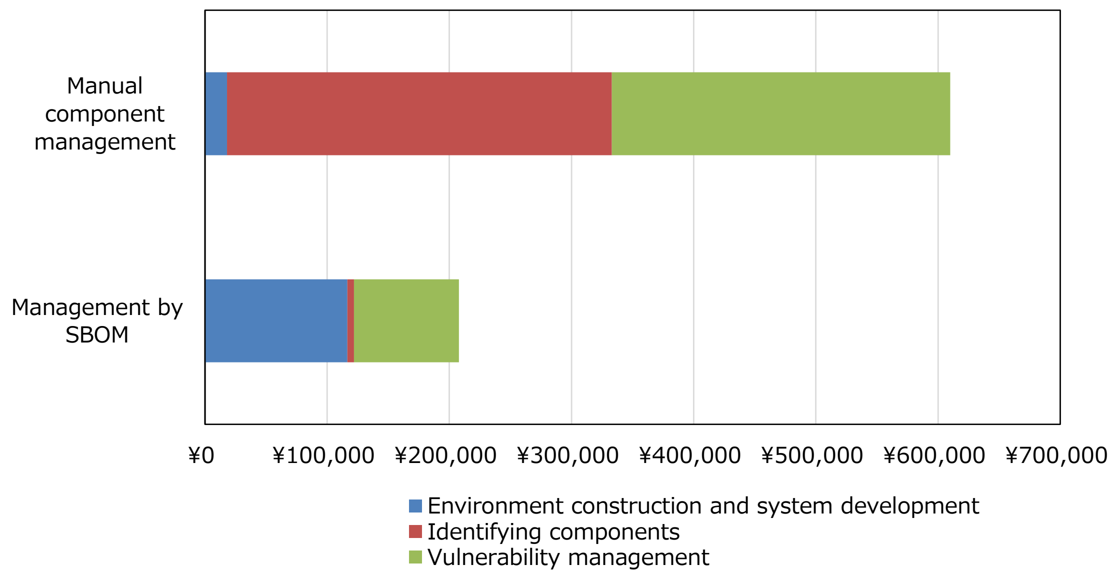


Figure 2-4 Results of reducing vulnerability management costs through the SBOM management (From the results of FY2022 PoC in the medical device sector)[9]

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 as a whole due to more products with fewer vulnerabilities.

[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.

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 2010, 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 measures according to the type of license. For example, in the case of the GPL, the GPL also applies to derivative works, and if new software is created by combining the GPL with other software, the GPL also applies to that software. 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, SBOMs 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 2021[10] and

[10] See Footnote 2.

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 SBOMs can facilitate software EOL management.

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 a large number of 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 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 also contributes to reducing the cost

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

of responding to vulnerabilities and identifying the impacted area.

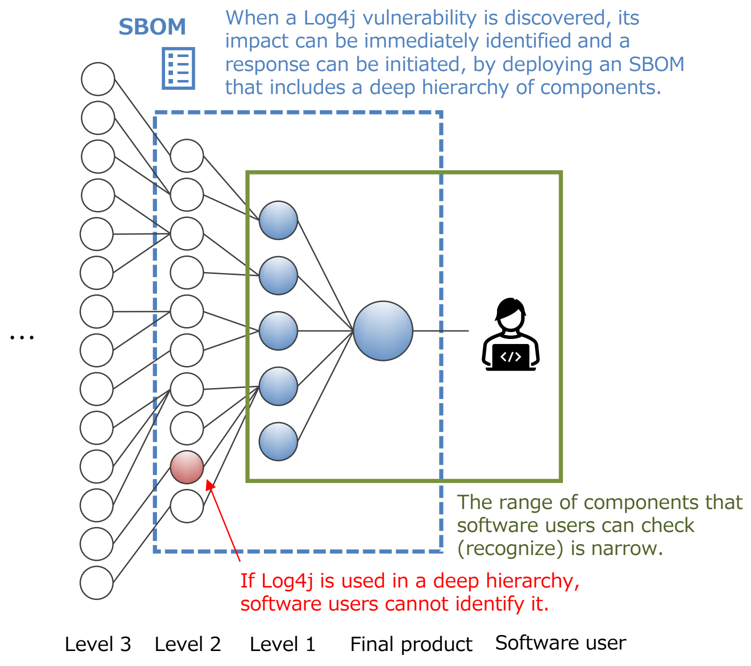

SBOM
When a Log4j vulnerability is discovered, its impact can be immediately identified and a response can be initiated, by deploying an SBOM that includes a deep hierarchy of components.
…
The range of components that software users can check (recognize) is narrow.
If Log4j is used in a deep hierarchy, software users cannot identify it.
Level 3
Level 2
Level 1
Final product
Software user

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 SBOM[12]. 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	This baseline component information includes: ● Supplier name ● Component name ● Version of the component ● Other unique identifier ● 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	Organizations utilizing SBOMs shall establish operational procedures for the following items: ● Frequency ● Depth[13] ● Known unknown[14] ● Distribution and delivery ● Access control ● Accommodation of mistakes[15]

[13]As 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

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

also increase interoperability when sharing SBOMs across organizations, thus contributing to transparency in the software supply chain. The following three formats are examples of SBOM format that can be used:

(1) SPDX（Software Package Data Exchange）

(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. SPDX items also include a Japan-originated format called SPDX-Lite, which contains only the minimum required items. 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 SBOMs 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 SBOMs created by Company A in different SBOM formats.

(1) SPDX（Software Package Data Exchange）

SPDX is an SBOM format developed by a project under the Linux Foundation, which was standardized as ISO/IEC 5962:2021 in September 2021. SBOMs 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 7.3.3(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. 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 #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	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-aefb-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"

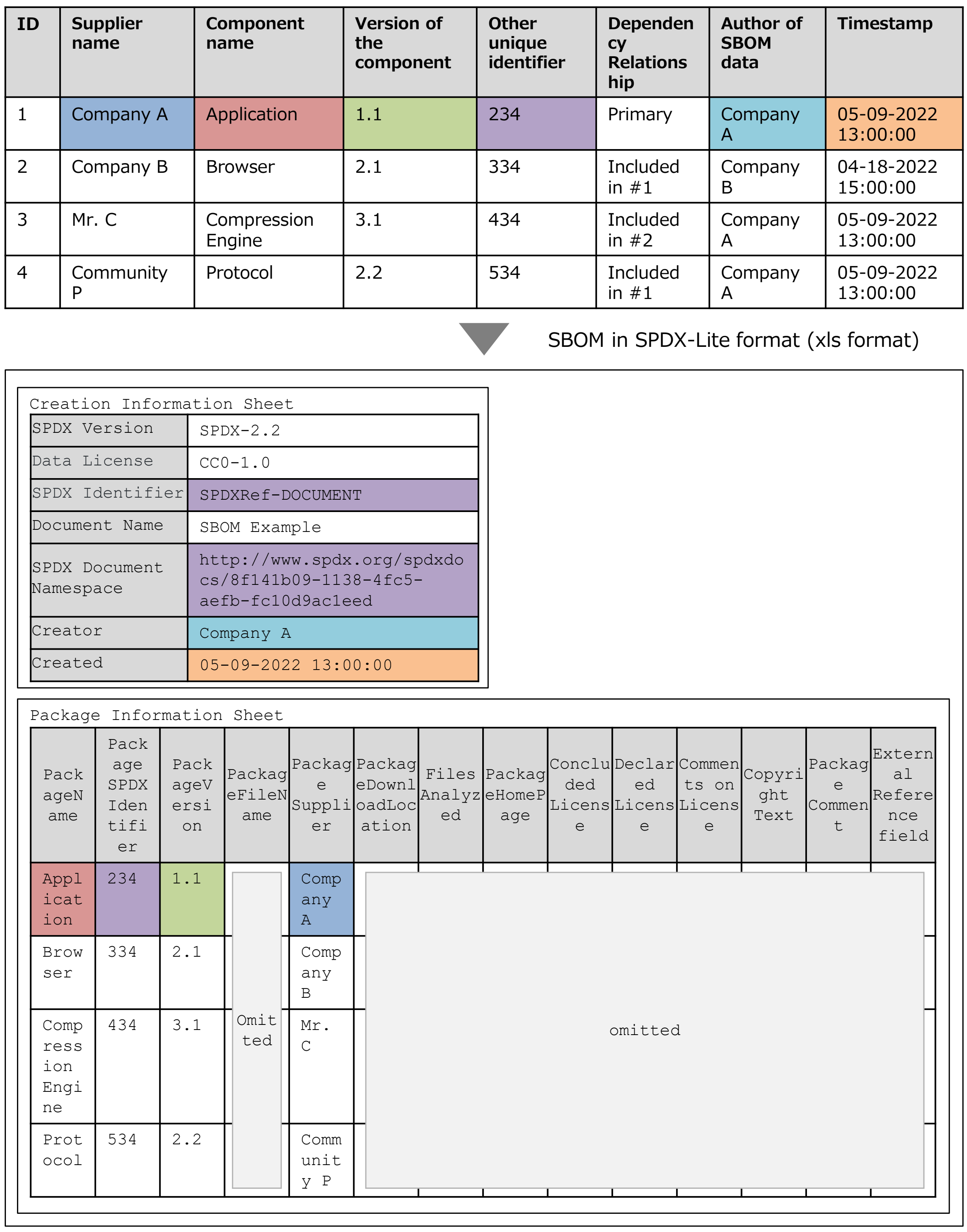
Data Fields of SBOM "Minimum Element"	Corresponding SPDX item
Supplier Name	PackageSupplier
Component Name	PackageName
Version of the Component	PackageVersion
Other Unique Identifiers	Combination of DocumentNamespace and SPDXID, ExternalRef

Data Fields of SBOM “Minimum Element”	Corresponding SPDX item
Dependency Relationship	Relationship （DESCRIBES; Representation by CONTAINS）
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 Japan-originated format called SPDX-Lite that includes the minimum required SPDX items. SPDX-Lite is designed for organizations that manually create license information and transfer only necessary information when SPDX-compliant 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 7.3.3(2) 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 #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	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

PackageName	PackageSPDXIdentifier	PackageVersion	PackageFileName	PackageSupplier	PackageDownloadLocation	FilesAnalyzed	PackageHomePage	ConcludedLicense	DeclaredLicense	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 or not to use SPDX-Lite. Furthermore, it should be noted that manual management of SPDX-Lite formatted SBOMs 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 7.3.3(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 #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	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
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 in order 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 #1	Company B	04-18-2022 15:00:00
3	Mr. C	Compression Engine	3.1	434	Included in #2	Company A	05-09-2022 13:00:00
4	Community P	Protocol	2.2	534	Included in #1	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:sha512="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

Data Fields of SBOM "Minimum Element"	Corresponding SWID tag item
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 SBOMs 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 SBOMs 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 SBOMs, 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. In particular, many commercial SBOM tools have a number of 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 2021[16] to clarify misconceptions and facts about SBOM. Below is a summary of the misconceptions and facts presented in the NTIA document.

Myth: SBOMs are a roadmap to the attacker

[Fact] Attackers can leverage the information contained in SBOMs. However, the defensive benefits of transparency far outweigh this common concern as SBOMs 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] SBOMs 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. The IP of third-party open-source components belongs to their respective authors or copyright holders. Also, the SBOM does not include any software

[16] NTIA, SBOM Myths vs. Facts
https://www.ntia.gov/files/ntia/publications/sbom_myths_vs_facts_nov2021.pdf

source code itself.

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

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

In addition, through PoCs 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 SBOMs, there may be cases where false positives or false negatives in the production of SBOMs, making it impossible to create accurate SBOMs. 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, SBOMs for online applications such as SBOMs for container images, SBOMs for SaaS software, and SBOMs for cloud services are also being discussed mainly in the U.S.

Myth: Only three formats of SBOM are allowed: SPDX, CycloneDX, and SWID tags; SBOMs 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 SBOMs 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 SBOMs, it is necessary to determine the scope of software for which to create SBOMs, as well as to clarify the issues that one's own organization wishes to solve by implementing SBOMs 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 SBOMs that include component dependencies, then it is expected that SBOMs will be created and managed using commercial SBOM tools. Also, for small-scale products that do not have a large number of 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 SBOMs 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 actually 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 chapters show the main implementation items for each phase and the points to note when introducing an SBOM.

Phase	Step	Outline of introduction
Environment and system development phase	Clarification the scope of the SBOM application	Clarify the scope of SBOM by organizing information on software subject to SBOM (language, contract type, regulatory requirements, internal constraints, etc.).
	SBOM tools selection	Organize the viewpoints for selection of SBOM tools and evaluate and select SBOM tools based on the viewpoints.
	SBOM tools installation	Install and configure the SBOM tool by reviewing the tool's instruction manual, README file, etc.
	Learning about SBOM tools	Learn how to use the SBOM tool by reviewing the tool's instruction manual, README file, etc.
▼		
SBOM production and sharing phase	Component analysis	Analyze the target software components and check the analysis results to determine if there are any false-positives and false-negatives.
	SBOM production	Determine the requirements for the SBOM to be produced, such as SBOM items, format, output file format, etc., and create an SBOM that satisfies such requirements.
	SBOM sharing	After considering how to share the SBOM to consumers and suppliers of the subject software, share the SBOM as necessary.
▼		
SBOM use and management phase	Vulnerability management and license management, etc.	Based on SBOM information on vulnerabilities and licensing, take appropriate vulnerability and license management actions.
	SBOM information management	Appropriately manage the information contained in the SBOM and the SBOM information itself.

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 chapter 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)	・ Produced internally ・ Produced by suppliers with business contract ・ Produced by suppliers without business contract (e.g., OSS community)
Timing of producing an SBOM (When)	・ During product planning or development planning ・ During program development ・ During software built ・ At software delivery ・ At component upgrading
Entity to use the SBOM (Who)	・ Software user ・ End-product vendor ・ Development vendor ・ End-product user
Scope of components covered by the SBOM (What, Where)	・ 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)	・ 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)	· 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)	· 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. In order 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 following[17].

- **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

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

- **Data formats handled by the organization**
  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. In particular, when SBOM tools are used to create SBOMs, 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. In order 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.

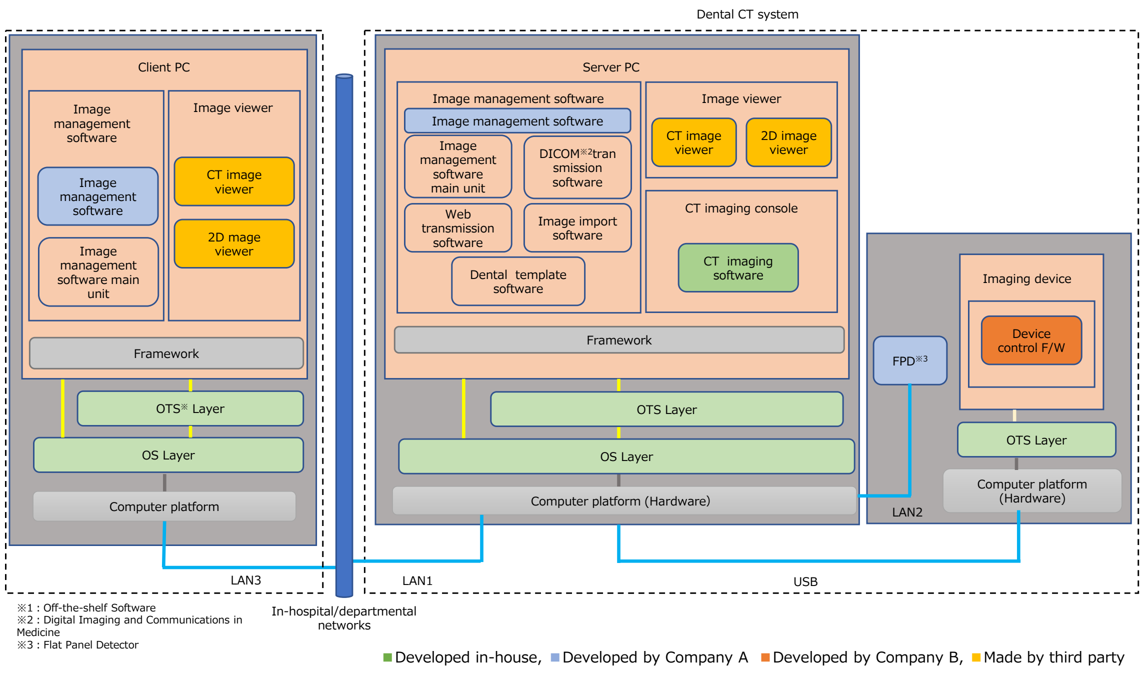

Dental CT system
Client PC
Image management software
Image management software
Image management software main unit
Image viewer
CT image viewer
2D mage viewer
Framework
OTS※ Layer
OS Layer
Computer platform
LAN3
Server PC
Image management software
Image management software
Image management software main unit
DICOM※2transmission software
Web transmission software
Image import software
Dental template software
Image viewer
CT image viewer
2D image viewer
CT imaging console
CT imaging software
Framework
OTS Layer
OS Layer
Computer platform (Hardware)
LAN1
USB
FPD※3
LAN2
Imaging device
Device control F/W
OTS Layer
Computer platform (Hardware)
In-hospital/departmental networks
※1 : Off-the-shelf Software
※2 : Digital Imaging and Communications in Medicine
※3 : Flat Panel Detector
■Developed in-house, ■Developed by Company A ■Developed by Company B, ■Made by third party

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

In addition, in order 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, in order 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 SBOMs is limited, but in the U.S., for example, software vendors that are subject to government procurement are encouraged to provide SBOMs[18]. In the EU, the Cyber Resilience Act, drafted in September 2022, includes SBOM requirements for digital products to be placed on the EU market[19]. 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 law and will be fully operational by the end of

[18] 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

[19] European Commission, Cyber Resilience Act https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act

2023. There is a possibility that SBOMs 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. In light of 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 take into account 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 SBOMs 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 a large number of 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 parts 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 SBOMs corresponding to the clarified scope of the

SBOM application. SBOM tools provide the most important facility for creating and managing SBOMs. 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 SBOMs, and this Guidance also assumes the use of SBOM tools.

Some SBOM tools are shown in 7.3.2 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 a large number of 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 parts 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[20] 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.

[20] 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	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 license fee is calculated in various ways depending on the number of developers, the scale of the organization, and the amount of analysis code. Depending on the tool, 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, taking into account the cost constraints within the company.
Supported formats	Depending on the SBOM tool, it is possible to import/create SBOMs 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.
Components analysis method	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.

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 the purpose of 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, taking into account 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 SBOMs 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 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[21]. 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,

[21] 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.

it is expected that multiple tools will be evaluated and selected based on 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 trial[22] 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

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

be achieved by obtaining assistance from companies that provide support 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 actually 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 SBOMs, 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 for the production of 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 SBOMs. 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 actually create SBOMs and provide them as needed. This chapter 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]

- SBOM tools can be used to analyze components and create SBOMs 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 an 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 PoCs confirmed that the SBOM tool can analyze components and create SBOMs 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 SBOMs using SBOM tools, and this section is also written assuming that SBOM tools will be used. In some cases, a package manager may 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, SBOMs can be created efficiently by receiving SBOMs 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 PoCs:

- 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 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 SBOM tool output results 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 PoCs, some components require 0.50 hours/component to check for false positives and false negatives, which means that a large number of workloads are required to check for false positives and false negatives in the case of software with a large number of components. In particular, checking for false positives and false negatives related to sub-tier components and third-party components requires a large number of 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 are and check if the detected components are included in the output results

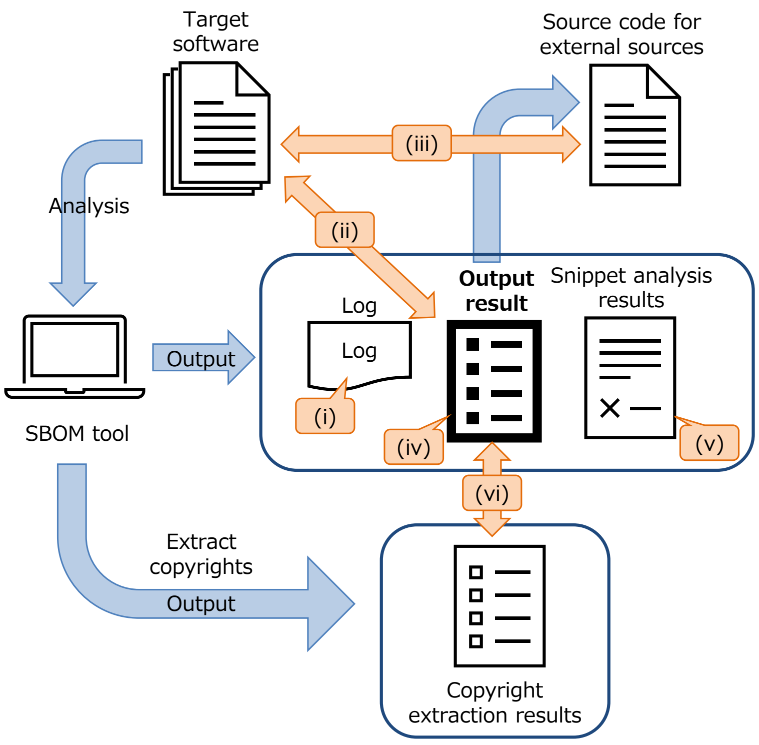


Figure 5-1 Perspectives and methods of checking component analysis results

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 taking into account, among other things, the possibility that a large number of 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, as a result 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 SBOMs from a third party, it is possible to create SBOMs efficiently, and organizations may also use them to examine the SBOMs created by the company. It should be noted that there are contractual and licensing issues regarding whether or not to request communities or individuals to provide SBOMs. In addition, if organizations are using a component provided by a third party after modifying it in their own organization, organizations should be careful because the provided SBOM cannot be used as it is. 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.

SBOMs should not only be created but also be managed continuously, and the date and time of creation of an SBOM should be clearly recorded. In order 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 SBOMs change dynamically after their creation due to the version-up of components. As described in Section 2.5, it is not mandatory to disclose SBOMs. SBOM creators and suppliers are encouraged to decide how to share SBOMs 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 SBOMs 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 SBPM sharing method, taking into account the characteristics and frequency of updates of SBOM target software, SBOM usage status among users, and so on. In addition, in order 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

In order to enjoy the benefits of SBOMs, it is required to operate and manage SBOMs that have created. This chapter 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.

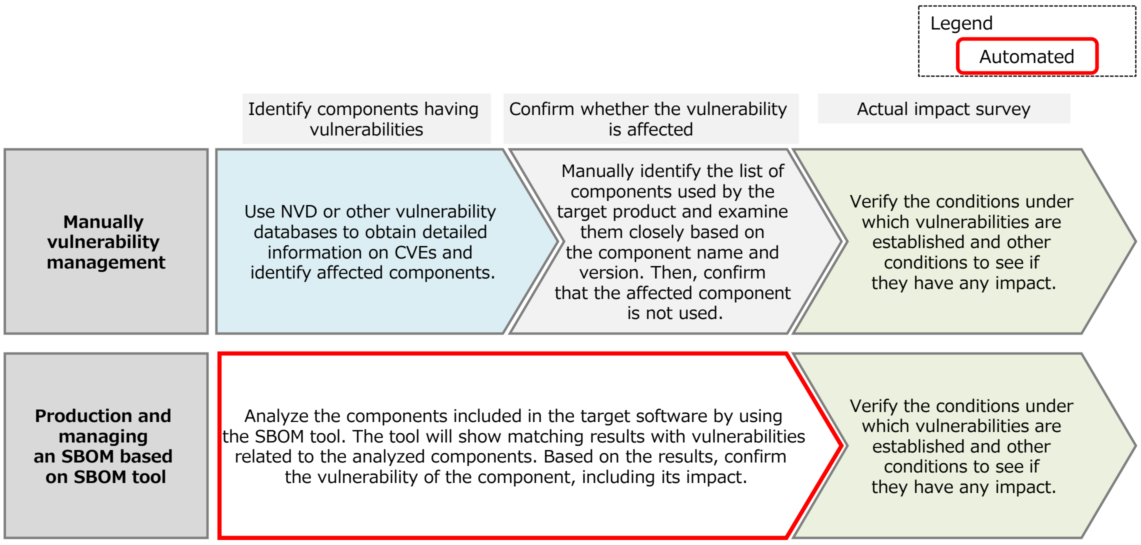
☐ Based on the output of the SBOM tool, check whether there is any violation of the OSS license.

[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, it is important to appropriately manage software using an SBOM. 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 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 make a decision 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 on a daily basis, 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.



Legend
Automated
Identify components having vulnerabilities
Confirm whether the vulnerability is affected
Actual impact survey
Manually vulnerability management
Use NVD or other vulnerability databases to obtain detailed information on CVEs and identify affected components.
Manually identify the list of components used by the target product and examine them closely based on the component name and version. Then, confirm that the affected component is not used.
Verify the conditions under which vulnerabilities are established and other conditions to see if they have any impact.
Production and managing an SBOM based on SBOM tool
Analyze the components included in the target software by using the SBOM tool. The tool will show matching results with vulnerabilities related to the analyzed components. Based on the results, confirm the vulnerability of the component, including its impact.
Verify the conditions under which vulnerabilities are established and other conditions to see if they have any impact.

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.

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 of time, 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.
- SBOMs 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 SBOMs shall be retained for a certain period of time, including a change history so that they can be referred to in case of inquiries from outside the company. The SBOMs should be retained for a minimum period of time while the product is generally distributed in the market. Even after the end of sales, it is necessary to maintain SBOMs for reference in advance because they may be referred to as necessary during the warranty period, support provision period, replacement parts 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 on a daily basis, 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 SBOMs, PSIRTs can reduce the workloads required for narrowing down the OSS actually 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, SBOMs 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 SBOMs as a deliverable and addressing vulnerabilities by utilizing SBOMs 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. Appendix

7.1. 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 7-1 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.	☐
		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.	☐
	SBOM tools selection	Organize the viewpoints for the selection of SBOM tools considering the development language of the target	☐

Phase	Step	Actions for the introduction of SBOM	Check
		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.)	
		Evaluate and select multiple SBOM tools based on the organized viewpoints.	☐
	SBOM tools installation	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.	☐
	Learning about SBOM tools	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.	☐
SBOM production and sharing phase	Component analysis	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.	☐

Phase	Step	Actions for the introduction of SBOM	Check
	SBOM production	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.	☐
	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.	☐
		Consider using electronic signature technology or other technologies to prevent falsification of the sharing of SBOM data.	☐
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.	☐
		Based on the output of the SBOM tool, check whether there is any violation of the OSS license.	☐
	SBOM information management	Keep the created SBOM for a certain period of time, including the change history, so that it can be referred to in case of inquiries from outside the company, etc.	☐
		Manage the information contained in the SBOM and the SBOM itself appropriately.	☐

7.2. Glossary

7.2.1. Terms related to SBOMs 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. SBOMs 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 SBOMs. 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 SBOMs.
- SBOM Tool
  A tool to produce, share, utilize, or manage SBOMs. 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.

7.2.2. 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]

7.3. Reference information

7.3.1. 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 SBOMs 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 SBOMs. 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 SBOMs and the needs of suppliers and users of SBOMs.
  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 Guide for SBOM Generation（October 2021）**
  A document summarizing two points of view as a guide for SBOM generation: how to collect information for collection method for SBOM generation and how to generate a specific SBOM. Although this guide 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_guide_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 SBOMs. This playbook summarizes the points to be considered when acquiring SBOMs from suppliers, the process and platform for utilizing SBOMs, and the intellectual property and confidentiality of SBOMs.
  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 SBOMs 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

7.3.2. SBOM Tools

This section shows some examples of SBOM tools that contribute to the creation, operation, and management of SBOMs. 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 particular 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.	· 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.	· 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	FOSSA	FOSSA, Inc.	· 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
4	FossID	FossID AB	· 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 SBOMs 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.
5	Insignary Clarity	Insignary, Inc.	· 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.
6	MEND SCA	WhiteSource Software, Inc.	· 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
7	Revenera SCA	Flexera Software LLC	· It enables the creation of accurate SBOMs, 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).
8	Snyk	Snyk, Ltd.	· 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.
9	Sonatype Lifecycle	Sonatype, Inc.	· 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.
10	Veracode SCA	Veracode, Inc.	· 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.

No.	Name	Developer	Features
11	yamory	Assured, Inc.	· 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 on a daily basis, 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	· It collects data on software repositories and normalizes them into a data model. · It collects data on OSS projects from many sources.
2	BOM Doctor	Sonatype, Inc.	· It generates SBOM by specifying a project URL or package URL on GitHub. · It visualizes generated SBOMs on a tree including dependencies of components (it is also possible to visualize SBOMs 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.	· 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.

No.	Name	Developer	Features
4	Daggerboard	NewYork-Presbyterian Hospital	· It provides a dashboard to view and manage SBOM and related vulnerabilities at a glance and can import SPDX or CycloneDX files for vulnerability detection.
5	Dependency-Track	OWASP Foundation	· 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	· It cannot identify the name and version of the OSS, but it is capable of detecting and managing the licenses and copyrights of the components included in the target software. · It allows import and analysis using a Web UI.
7	in-toto	Linux Foundation	· 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	OSS Review Toolkit (ORT)	Linux Foundation	· 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.
9	SBOM Tool	Microsoft Corporation	· It integrates with various package management systems such as NPM, NuGet, PyPI, etc. to automatically detect and create SBOMs in SPDX format. · It runs on Windows, Linux, and macOS platforms.

No.	Name	Developer	Features
10	ScanCode .io	nexB, Inc.	· 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.
11	Scancode Toolkit	nexB, Inc.	· 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.
12	SW360	Eclipse Foundation	· It identifies and manages security vulnerability information for software components. · It identifies and manages license information for software components.
13	SwiftBOM	CERT Coordination Center (CERT/CC)	· It allows manual input for SBOM generation. · It imports previously created SBOMs and displays SBOMs in a tree-like view.
14	Syft & Grype	Anchore Enterprise	· 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.
15	Trivy	Aqua Security Software, Ltd.	· 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.

7.3.3. SBOM data formats

The SBOM “Minimum Elements” include a category of “Automation Support”, which takes into account support for automation in the automatic generation, readability of SBOMs, etc. As specific data formats, three formats—SPDX (Software Package Data Exchange), 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 SBOMs 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,[23] and the project continues to study and update it. 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

SBOMs in the SPDX format contain information about components created according to the SPDX Specification, license, and copyright. Tag:Value(txt), RDF[24], XLS, JSON[25], YAML[26], and XML[27] formats are supported. Sections and items 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

[23] https://spdx.GitHub.io/spdx-spec/v2.3/

[24] 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.

[25] 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.

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

[27] 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.

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 SBOMs 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 website[28] 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 7-2 Relationship between SPDX-Lite items and SPDX

Section name in SPDX	Item name in SPDX-Lit
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
	Package Home Page
	Concluded License
	Declared License
	Comments on License
	Copyright Text

[28] https://spdx.GitHub.io/spdx-spec/v2.3/SPDX-Lite/

Section name in SPDX	Item name in SPDX-Lit
	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 SBOMs 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 SBOMs 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 site[29] and are being maintained and updated by the core working group of the OWASP community. As

[29] https://cyclonedx.org/docs/1.4/json/

an overview of the CycloneDX v1.4, the following provides the structure of the format, examples of use and purpose of the format, and features of the format.

1) Format configuration

SBOMs 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 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 in order 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 SBOMs 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.