Nationaal Cyber Security Centrum
Ministerie van Justitie en Veiligheid

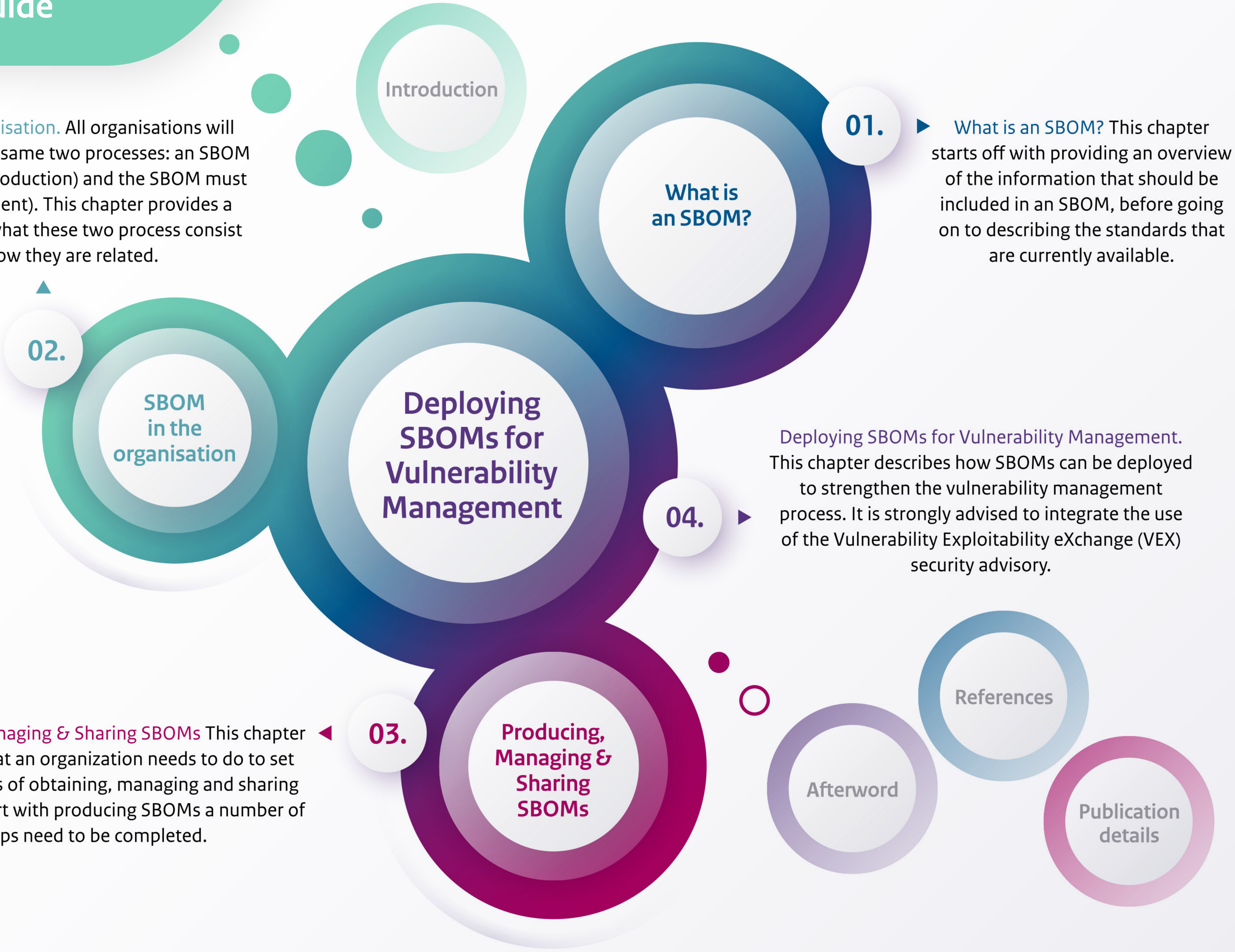
Software Bill of Materials (SBOM)

What, why and how?

Starter guide

TNO innovation for life

Use this figure to browse through this start guide



SBOM in an organisation. All organisations will have to arrange the same two processes: an SBOM must be created (production) and the SBOM must be used (deployment). This chapter provides a brief overview of what these two process consist of and how they are related.

What is an SBOM? This chapter starts off with providing an overview of the information that should be included in an SBOM, before going on to describing the standards that are currently available.

Deploying SBOMs for Vulnerability Management. This chapter describes how SBOMs can be deployed to strengthen the vulnerability management process. It is strongly advised to integrate the use of the Vulnerability Exploitability eXchange (VEX) security advisory.

Producing, Managing & Sharing SBOMs This chapter describes what an organization needs to do to set up the process of obtaining, managing and sharing SBOMs. To start with producing SBOMs a number of steps need to be completed.

Introduction

Over the past few years, a succession of major supply chain incidents such as SolarWinds and Log4J have made it painfully apparent that many organisations have insufficient understanding of the dependencies within their software supply chain [1↗]. Software Bill of Materials (SBOM) is an important building block in tackling this problem. By using SBOMs, organisations can maintain a formalised and permanently up to date overview of all the software used and their dependencies.

Insight into the software supply chain can serve various purposes within organisations. Among others, the previous studies by Capgemini [2↗] and by the NTIA [3↗] provide a detailed overview. The two main reasons for organisations to work on their SBOMs *right now* are

- **Vulnerability Management**: creating the automated link between public vulnerabilities and vulnerable products, and
- **Compliance Management**: demonstrably complying with statutory and other requirements that offer insight into software dependencies.

In this guide, the focus is on the use of SBOM for vulnerability management, because that is the area in which SBOM delivers the greatest added value. However, from this perspective it is also important to focus attention on developments in the field of legislation and regulations so that these can be included in reaching agreements and selecting tooling. The detail block on this page briefly discusses the most important developments.

Despite the fact that SBOM is widely talked about, the term still raises many questions. It is often unclear to organisations precisely what SBOM can mean, what is required to successfully integrate SBOM and how SBOM relates to other elements such as the Vulnerability Exploitability eXchange (VEX) and the Common Security Advisory Framework (CSAF) This starter guide was developed to assist managers who are involved in the

Close detail block

Detail block:
Legislation and regulations relating to SBOM

Since 2021, the Executive Order on Improving the Nation's Cybersecurity [19↗] requires software supplied to the US Federal Government to include an SBOM. The EU is working on a similar regulation: the Cyber Resilience Act (CRA) [18↗]. This Act is expected to be approved in mid-2023 and will make it compulsory to generate SBOMs for software components in products with digital elements, within the EU. In addition, sector-specific agreements are currently being reached within various sectors (for example healthcare and automotive) regarding the use of SBOMs. Finally, the importance of SBOMs is also explicitly referred to in the ISO/IEC 27036 standaard *Cybersecurity – Supplier relationships* currently being developed.

At present, it is expected to be several more years before organisations actually must comply with the CRA. Nevertheless, experts at Synopsys [15↗] already advise software suppliers to check which requirements their SBOMs will have to comply with in the future. The development of a software product is often a multiyear process and an understanding of the requirements could represent an important factor in selecting tools or equipping processes.

with
nto

lity
vers
t is
field of

ck on

n still

sfully
such as

between public vulnerabilities and vulnerable products, and

the Vulnerability Exploitability eXchange (VEX) and the Common Security Advisory Framework (CSAF) This starter guide was developed to assist managers who are involved in the

Intro

cybersecurity of their organisation to understand these issues. In this guide, NCSC-NL and TNO provide answers to frequently asked questions and offer clear recommendations for starting with SBOM.

The information in this guide is based on desk research, a workshop with representatives of the target group for this guide and interviews with organisations that have already gained experience with the use of SBOM: Philips, ABB and Siemens. A discussion was also held with Allan Friedman, representative of the American Cybersecurity and Infrastructure Security Agency (CISA). In the research, sources made available by the American National Telecommunications Information Administration (NTIA) were regularly consulted. Finally, this guide builds on the results of a previous study by the NCSC and Capgemini: Using the Software Bill of Materials for Enhancing Cybersecurity [2↗].

Back to starting figure

Intro

Reader's guide

The next chapter *What is an SBOM*? offers a short description of what an SBOM looks like and what choices have to be made regarding the SBOM. The next chapter *SBOM in the organisation* provides an overview of the important steps that must be taken when using SBOMs for vulnerability management. These steps are further elaborated in the subsequent chapters. The process starts with acquiring and/or supplying SBOMs to internal or external parties in the chapter *Producing, Managing & Sharing SBOMs*. The next chapter describes how these SBOMs can be deployed for vulnerability management under the heading *Deploying SBOMs for Vulnerability Management*. The detail blocks that appear throughout this document provide additional information about specific subjects, including links to external sources.

What is an SBOM?

What is an SBOM?

01.

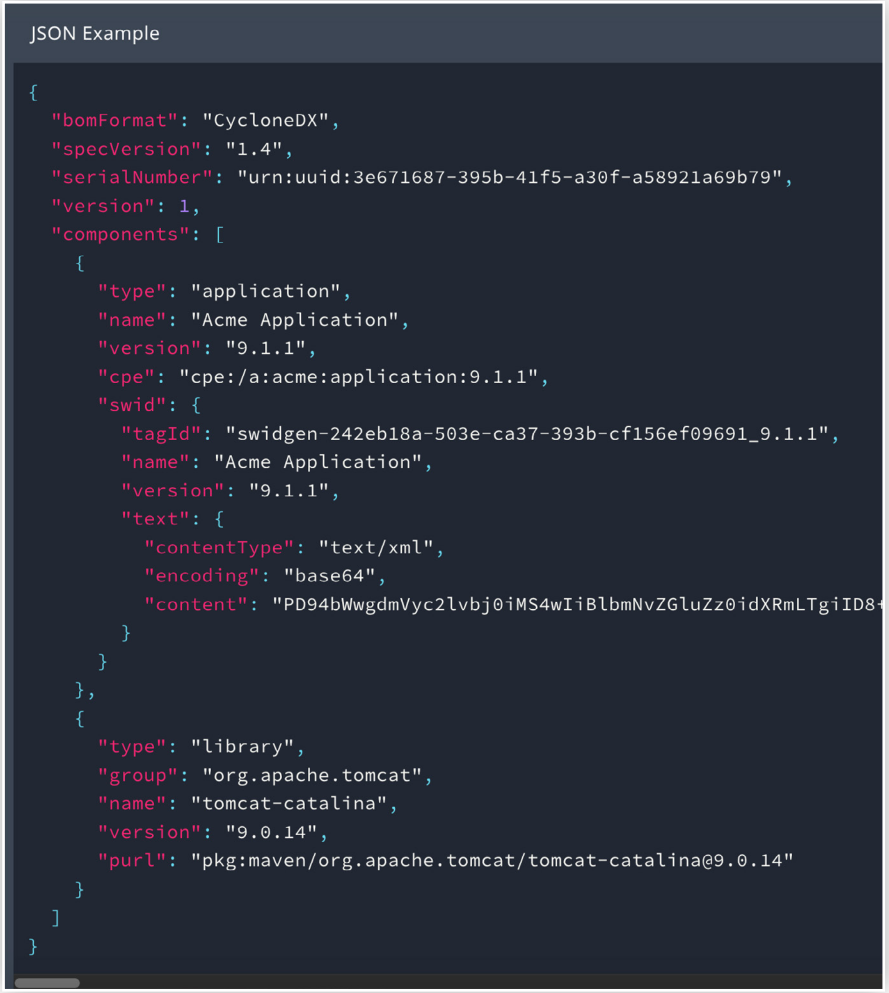
Back to starting figure

What is an SBOM?

An SBOM describes the components that make up a piece of software and the relationships between those components. It is a nested list of software components comparable to the list of ingredients on food products. One vital aspect of this list is that the structure is formalised (i.e. machine readable) thereby opening the way for automation. **Figure 1** is an example of a simple SBOM. The concept of a component list is neither new nor specific to software. Alongside the SBOM, in certain sectors a Hardware Bill of Materials (HBOM) or Component Bill of Materials (CBOM) are also used. These two can also play a role in the framework of vulnerability management and supply chain security, but are beyond the scope of this starter guide.

This chapter starts by providing an overview of the information that should be recorded in an SBOM, before considering which standards are currently available.

Figure 1: Example of an SBOM | *Source: CycloneDX Use Cases*



JSON Example

```json
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.4",
  "serialNumber": "urn:uuid:3e671687-395b-41f5-a30f-a58921a69b79",
  "version": 1,
  "components": [
    {
      "type": "application",
      "name": "Acme Application",
      "version": "9.1.1",
      "cpe": "cpe:/a:acme:application:9.1.1",
      "swid": {
        "tagId": "swidgen-242eb18a-503e-ca37-393b-cf156ef09691_9.1.1",
        "name": "Acme Application",
        "version": "9.1.1",
        "text": {
          "contentType": "text/xml",
          "encoding": "base64",
          "content": "PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiID8
        }
      }
    },
    {
      "type": "library",
      "group": "org.apache.tomcat",
      "name": "tomcat-catalina",
      "version": "9.0.14",
      "purl": "pkg:maven/org.apache.tomcat/tomcat-catalina@9.0.14"
    }
  ]
}
```

What is in an SBOM?

What is an SBOM?

01.

Back to starting figure

What is in an SBOM?

At the time of writing, there is no unequivocal description of exactly what an SBOM should look like, and which elements an SBOM must contain. There are different standards, each with their own structure and mandatory fields. The various standardisation and regulation initiatives have also only revealed a limited number of requirements. The update to the ISO/IEC 27036-3[1] standard currently being developed describes a set of essential elements that should appear in an SBOM. This set is more detailed than the minimum set of requirements specified by the American government [4↗] but does contain a number of elements which increase the possibility for effectively deploying SBOMs. The essential elements are:

- **Author**, the person/organisation that generated the SBOM
- **Timestamp**, for the production or revision of the SBOM
- **Life cycle**, where in the software development process the SBOM was generated (pre-build, build and post-build)
- For each component
  - **Supplier name**, the person/organisation supplying the comp
  - **Component name**, this can be described in a number of different ways
  - **Version**
  - **Cryptographic hash**, hash of the component which among others can be used for identifying the component
  - **Unique identifier**
  - **Relationships**, the relationships between components
  - **Source**, where the component was obtained (e.g. GitHub or a specific package manager)

The *unique identifier* element is of real importance in the (automated) use of SBOMs, but here too there is no uniform standard available as yet. The **detail block on this page** describes this problem more precisely. In practice, this means that when reaching agreements on the content of an SBOM, specific attention must also be paid to how the *unique identifier* element will be implemented.

1 At the time of writing, this standard is still in the review process and has the status *Final Draft International Standard* (FDIS).

SBOM Standards →

Close detail block

Detail block:
Unique identification for software

To enable the effective use of SBOMs, it is essential that software components are identified in the same way, everywhere. Only then a software componant from an SBOM can be automatically linked to known vulnerabilities, licences and other relevant information. If this does not happen, *false positives* or *false negatives* can arise. In the case of a *false positive*, a match is made between information that is not actually related. If this happens when searching for vulnerabilities, a component can wrongly be identified as vulnerable. In the case of a *false negative*, actually no match is made where it should be. This means that a component may appear to not be vulnerable, while in fact it is. Unfortunately, at present, too many different ways of identifying software components are in use. The most commonly used methods are:

- Coordinates
- Package URL (PURL)
- Common Platform Enumeration (CPE)
- Software Identification (SWID) tagging
- Cryptographic hash functions (SHA-1, SHA-2, SHA-3, BLAKE2b, BLAKE3)

The use of **coordinates** is described as: *group*, *name* and *version*. An example is "group": "org.example", "name": "sample-library", "version": "1.0.0.

PURL was the result of an initiative to introduce a standard method of identifying software packages, independently of the context of a specific package manager. It makes use of a simple syntax in which a number of components are combined to form a URL: scheme:type/namespace/name@version?qualifiers#subpath. This results in identifiers such as: pkg:maven/org.apache.xmlgraphics/batik-anim@1.9.1?packaging=sources.

CPE is a commonly used standard, in particular in combination with vulnerability and incident management. However, the process surrounding the setting of a CPE is unfortunately vulnerable to human error. In addition, the granularity of the CPE approach is too limited in certain situations to determine whether a software component actually contains a vulnerability. For that reason, the SBOM Forum recommends switching to the use of PURL [17↗].

1 2 →

SBOM Standards →

Close detail block

The **Software Identification** (SWID) Tag is an ISO-standaard [16↗] which was already defined in 2012 as a means of maintaining an overview of the software installed within an organisation. In practice, in the context of SBOM, SWID Tags are primarily used as a means of identifying software components within the CyclonDX or SPDX format, which is explained in the section on *SBOM Standards*.

The use of **hashes** is a well-known means of verifying that two pieces of code are identical. In practice, this proves difficult to use because different hash functions are used or because it is unclear precisely which components of a package have been included in the hash. For example, is the SBOM itself part of the hash if delivered as part of the package, or is it not?

In addition to the use of different standards, the names used by suppliers for software are also subject to changes, for example as a result of mergers and acquisitions between the organisations, or as a result of new project forks.

The conclusion is that there is still a long way to go before a uniform global approach will be introduced for identifying software components. Until that time, the NTIA recommends applying the following principles [14↗]:

- If a uniform method already exists for identifying a component, follow this method. Examples are names from package managers that use unique identifiers or names from suppliers that allocate clear identifiers to their software
- If no uniform method exists yet, choose an existing, commonly used standard to describe a component.

1 2

SBOM Standards

Back to starting figure

What is an SBOM? 01.

SBOM Standards

At the time of writing this guide, there are two widely used SBOM standards [5↗] :

- CyclonDX, a lightweight open-source standard managed by the CyclonDX working group, which had its roots in the Open Worldwide Application Security Project (OWASP) community;
- Software Package Data Exchange (SPDX), an ISO-certified open-source standard (ISO/IEC 5962:2021) managed by the Linux Foundation Project.

As yet, there is no consensus within the SBOM community on which standard can best be chosen. In practice, the standards are interoperable because there is sufficient overlap between the different fields and the essential elements can be described in all standards [5↗]. The best practice is to support both SPDX and CyclonDX. There are tools that can handle both formats and tools that can convert one format into the other. This may result in the loss of additional information, so it is important to investigate in advance which information is important for your own organisation.

SBOM in the organisation →

SBOM in the organisation

02.

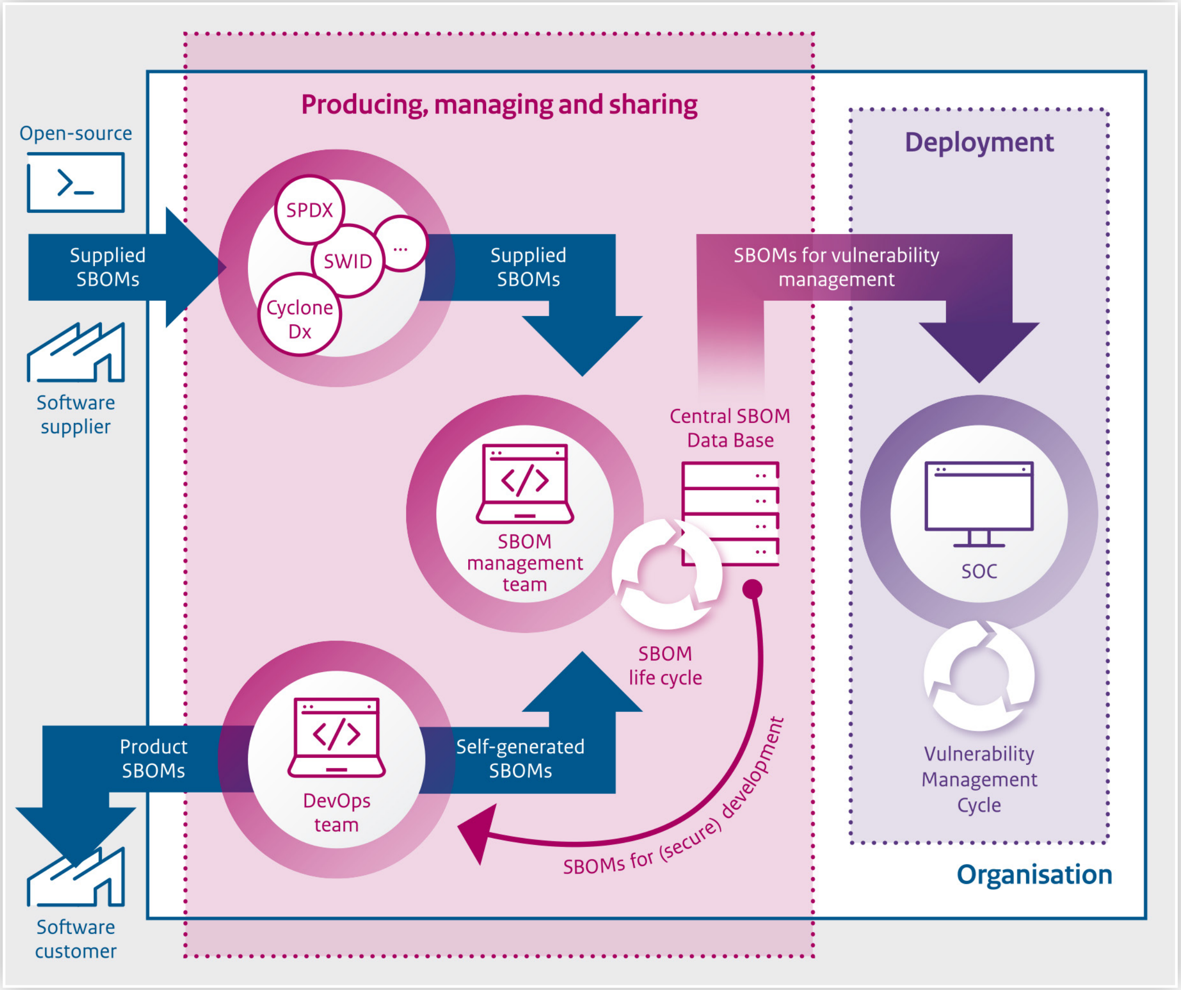
Back to starting figure

SBOM in an organisation

Now the content of an SBOM is clear, the next question is: how can I use it in my organisation? To eventually be able to use SBOMs for vulnerability management, a number of processes must be set up.

Figure 2 provides a diagrammatic overview of the use of SBOMs for vulnerability management within an organisation and the various steps that must be taken. This chapter describes these steps in brief, and the details are further elaborated in the next two chapters.

Figure 2: Overview of SBOM processes within the organisation



SBOM processes

SBOM in the organisation

Back to starting figure

SBOM in an organisation

SBOM processes ▶

SBOM processes

Precisely which processes need to be equipped to implement SBOM depends heavily on the organisation-specific context. Nevertheless, broadly speaking, all organisations will have to organise the same two processes:

- **Obtaining the necessary information**, which includes the production, management and eventual resharing of SBOMs.
- **The deployment of the information obtained**, which within the context of this guide is limited to the deployment of SBOMs for vulnerability management.

SBOMs can be obtained in two ways:

- The SBOM is provided by the software supplier. This may either be a commercial software supplier or an open-source community.
- The SBOM must be generated by the organisation itself. This is necessary for software developed in-house, or for software for which no SBOM is (currently) available such as legacy software or open-source code. For software currently under development, this process will often be carried out by development teams, but for legacy software, the task could be entrusted to a different team.

Many organisations will obtain their SBOMs via both routes. After an SBOM has been produced, it must be made available within the organisation. This sometimes requires a processing stage in order to retrieve the relevant information from the supplied SBOM. Ideally, SBOMs are then made available at a central location for all parties that have to use it. In the case of vulnerability management, this will often be the Security Operations Center (SOC). Because the software or configuration can be updated, any changes must be recorded in a new SBOM. Ensuring that each version of the software and the configuration has the appropriate SBOM is therefore an important aspect of integrating SBOMs in the organisation. If the organisation supplies software to third parties, the relevant SBOMs must also be shared with those parties. Here too, a processing stage may be necessary to determine which information needs to be shared with which parties.

SBOM in the organisation

02.

Back to starting figure

To effectively deploy SBOMs for vulnerability management, the recommendation is to integrate this use with already existing processes as far as possible. In these processes, SBOM will above all be useful in creating an overview and insight more quickly. This information can then be used in making risk assessments and arriving at mitigating measures.

SBOM roles

Because SBOM affects so many aspects of an organisation, different departments and roles within the organisation will be involved in the production or use of an SBOM [2↗]. Departments such as purchasing, contract management and legal will often be responsible for reaching agreements with external parties regarding the receipt and delivery of SBOMs. Departments such as IT management and development will play a driving role in generating SBOMs and keeping them up to date, while the security teams will be the most important users when it comes to deploying SBOMs for vulnerability management. Agreeing on the parameters for successfully matching up these processes is essential. It is therefore advisable to reach clear agreements on how and when the various teams will be involved, for example using RACI tables.

Producing, Managing & Sharing SBOMs

03.

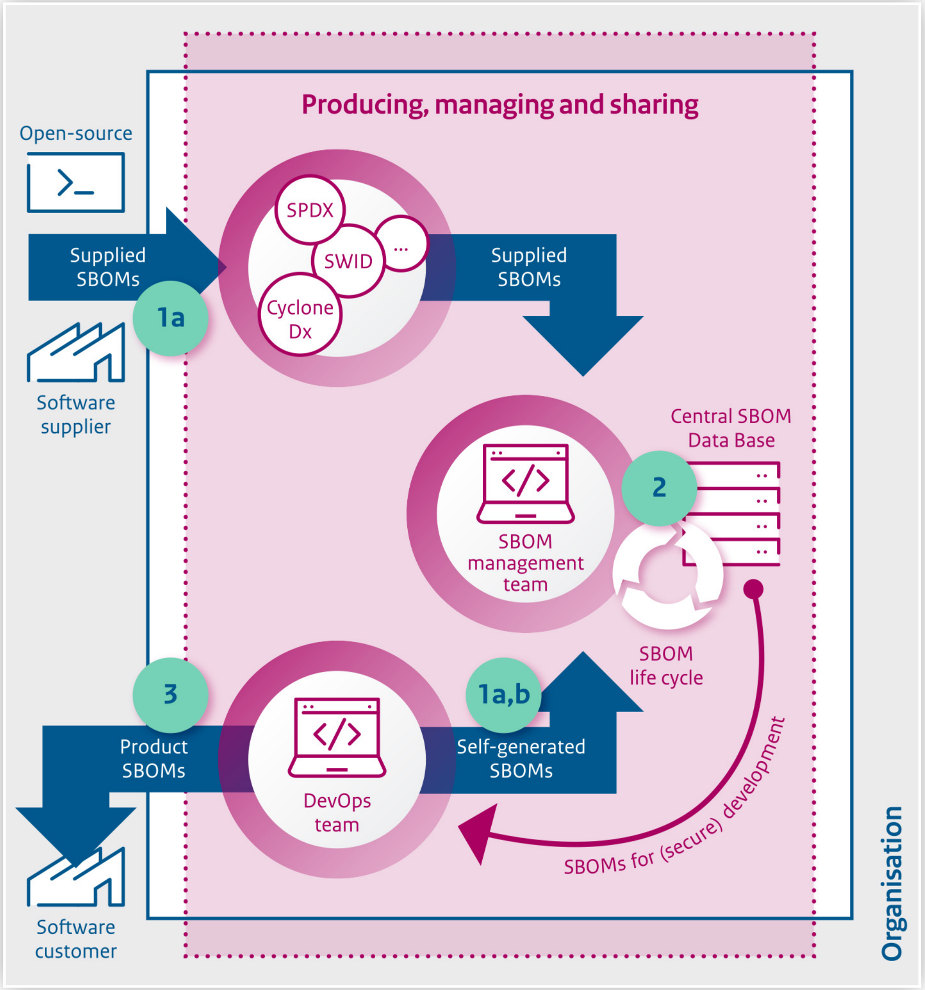
Back to starting figure

Producing, Managing & Sharing SBOMs

This chapter describes what an organisation needs to do in order to successfully set up the process of obtaining, managing and sharing (making available) SBOMs. Before you start with the task of producing SBOMs, a number of steps must first be completed. The various steps are briefly listed below and are further elaborated in a separate section.

1. Arrange the processes for obtaining SBOMs:
   - Reach clear agreements with the parties (internal or external) supplying the SBOMs.
   - If relevant, organise the generation of SBOMs for software developed in-house and open-source software.
2. Organise a process (including the necessary tooling) for processing and managing the received SBOMs.
3. Organise the distribution of SBOMs to third parties.
4. Evaluate the previous steps and make adjustments where necessary.

▼ *Figure 3: Overview of the processes within the production, management and sharing of SBOMs*



Go to step 1 ➔

Producing, Managing & Sharing SBOMs

03.

Back to starting figure

Step 1.
Arrange the processes for obtaining SBOMs

Stap 1a. Reach agreements with SBOM suppliers

Entering into dialogue with (external) software suppliers about also supplying SBOMs is an important step in working with SBOM. Even if the deployment process for SBOMs has not yet been fully arranged within your own organisation, it is still meaningful to start these conversations. By initiating the dialogue at this stage, it is possible to facilitate a phased approach in which the agreements between the parties can be further focused on the basis of experience acquired.

If the supplier offers cloud-based software or software as a service (SaaS) this will also complicate supplying an SBOM. In that case, the software is not operated on the customer infrastructure, and because software versions for these products are generally more often subject to changes, it may prove impractical to also supply SBOMs. For that reason, the NTIA recommends that in the near future suppliers must have internal SBOMs, and that they must act on the SBOM information in a timely manner, but that SBOMs for cloud-based software and SaaS applications need not be shared [4↗]. As SBOMs become more developed, the information from these SBOMs could also be included in the customer risk-management strategy.

The SBOM standards and legislation and regulations currently available offer a great deal of freedom regarding the information that must be stored in an SBOM, and the processes surrounding the exchange of SBOMs. It is therefore important to reach clear agreements with suppliers. These agreements should at least contain the following elements [4↗]:

- Format of the SBOMs
- Method of supply
- Methods for guaranteeing the authenticity of SBOMs, for example via digital signatures
- Frequency of updates
- Fields/information to be completed
- Depth of the SBOM (see also the **detail block: Quality of SBOMs** ❐)
- How to deal with missing information
- What are the obligations upon the supplier with regard to identified vulnerabilities

1 2 3 →

Producing, Managing & Sharing SBOMs

03.

If the SBOM must be provided by an external supplier, it can be useful to not reach these agreements on a one-to-one basis because both for the supplier and the customers, this involves a great deal of work. If customers with comparable information needs join forces (for example in the form of an ISAC), a better level of harmonisation could be achieved with the SBOM suppliers and the suppliers themselves will be more easily able to satisfy the specific needs of the consumer. In such discussions, the advice is to not start from scratch but to keep pace with the standards currently under development such as the ISO/IEC *27036: Cybersecurity – Supplier relationships*.
Finally, it is expected that in the future legislation and regulations will impose more specific requirements on the form and content of SBOMs. The Cyber Resilience Act for example states that the European Commission can impose further requirements on SBOMs in the form of implementing legislation (Article 10, section 15). It is therefore important to keep monitoring these developments.

Step 1b: Organise the generation of SBOMs

Depending on how the processes within an organisation are structured, SBOMs will be generated at different moments or by different roles. The specific way in which the implementation process will be carried out will therefore largely depend on the organisation. In broad terms, the generation of an SBOM involves four stages [6↗]:

- Identify the software components used
- Gather the necessary data about the software components
- Integrate the gathered information in the chosen SBOM format
- Check the resultant SBOM

In practice, stages one to three are often viewed as a single large stage. Ideally, these stages will be implemented as a fully automated process, as an integral part of the software development process. This will prevent human errors and inconsistencies. Furthermore, the generation of SBOMs during the development of software can itself contribute to the development of better and more secure products, for example by including information about preferred software components (for more information see the chapter on SecDevOps in [2↗]).
Depending on the software building platform used, different tools are available that are capable of automatically generating an SBOM during the software building phase. No specific tools are mentioned in this guide, but on the websites of SPDX and

Producing, Managing & Sharing SBOMs

03.

CycloneDX lists are kept of the tools that work with the standards in question.

There are a number of aspects which must be taken into account when selecting a suitable tool:

- How does the tool tie in with the current software development process and the existing tool? Is there perhaps a tool available which can be directly integrated in the current development environment or test environment?
- On which types of file must be tool be usable; is the source code always available or must the tool for example also work on Docker images?
- How does the tool handle nested dependencies; what is the maximum step depth the tool can reach? (See also **the detail block on the quality of SBOMs**)

In certain cases, it is not possible to generate an SBOM during the development of the software (for example for legacy systems). In these cases, the SBOM can be produced post-build. A post-build SBOM can be generated on the basis of SBOMs for sub-components or on the basis of scans of the software using code analysis tools. Additional manual work will often still have to be carried out, for example to retrieve information from licence agreements.
Two important points for attention when generating SBOMs are the inclusion of runtime dependencies and container information. Today, a lot of software is created with the assistance of containers like Docker. To prevent the generation of false negatives during the automated matching of information with the components of an SBOM, the SBOM itself must also contain information about these runtime dependencies and all containers.

For stage 4, checking the generated SBOM, it is possible to check whether the layout of the SBOM conforms with the requirements from the standard. Tools are available for this purpose for the various standards (see also the websites of SPDX and CycloneDX). To verify the content of the SBOM (a task for the supplier or the consumer), use can be made of frameworks that evaluate the maturity of the SBOM generation process and are consequently able to allocate a specific degree of reliability to the resultant SBOM [6↗]. Two examples are: the OWASP Software Component Verification Standard (SCVS) and ISO 5230.

Pro
& S

Producing,
Sharing SB

Step 1.
Arrange t
for obtai

Step 2.
Organise th
manageme

Step 3.
Organise th
SBOMs to t

Step 4.
Evaluate an

Close detail block ✕

Detail block:
The quality of SBOMs

Because no consensus has yet been reached about exactly what must be recorded in an SBOM, there is also no standardised approach to evaluating the quality of an SBOM. Best practice is to have SBOMs generated automatically during the building process as much as possible. In this way, human errors (for example in the processing of naming and updating) are avoided, as much as possible. Due to the wide variety of tools and configuration options, even in this situation, it is not self-evident that an automatically generated SBOM will comply with (implicitly) expected quality requirements. In addition to the continuous ever present challenges relating to the unique identification of components, there are two other important points for attention in evaluating the quality of an SBOM:

- Depth
- Completeness

The **depth** of an SBOM refers to the number of stages in which dependencies on dependencies (i.e. transitive dependencies) are included. The minimum requirements presented by the American government state that an SBOM must at least describe the primary components with all direct dependencies [4↗], in other words, 1 step. These components and dependencies must be described in sufficient detail to be able to iteratively identify the transitive dependencies. For many SBOM applications, it is essential to establish the most complete possible overview of the software supply chain such that the number of steps should be as high as possible. A good example of the importance of >1 step is described in [3↗] . At present, it can prove difficult for software suppliers to create deep SBOMs, due to existing requirements imposed by sub-component suppliers. The expectation is that as more organisations start to

1 2 →

Back to starting figure

← 1 2 3

Go to step 2 →

Close detail block ✕

work with SBOMs, it will become possible to add more detail. At that point, the depth could also become a quality requirement recorded in the agreements between the software supplier and the consumer.

When analysing the **completeness** of an SBOM, it is particularly important to be able to distinguish between components that have no dependencies and components of which the dependencies are unknown (i.e. *known unknowns*). Determining the completeness of an SBOM is not a trivial task. To be able to evaluate a generated SBOM, knowledge must be available about the software for which the SBOM was generated. This may sound paradoxical, but it does mean that SBOM suppliers must have a thorough understanding of their configuration management if they are to be able to validate an SBOM. For organisations that consume SBOMs, certainly in the early stages of using SBOMs, this evaluation process will be even more difficult.

In other words, at present it is difficult to evaluate the quality of SBOMs. It is therefore important both for SBOM suppliers and SBOM consumers that a process is designed to make it possible to identify non-conformities and shortcomings, and hence to subsequently improve the quality of SBOMs. Certainly during the early phases of use of SBOMs, this is expected to be a manual and knowledge-intensive process.

← 1 2

← 1 2 3 Go to step 2 →

Producing, Managing & Sharing SBOMs

03.

Step 2:
Organise the processing and management of SBOMs

To be able to make use of SBOMs within an organisation, it is important that the relevant information from all SBOMs is made available in a uniform manner. Ideally, this information will be read out and stored automatically in a central database. For an ever growing number of applications, tools will become available for reading SBOMs directly into a standard format (CyclonDX, SPDX) However, this will not immediately be possible for all SBOMs and additional processing stages will have to be organised.

Two example are [7↗]:

- Due to the challenges relating to the unique identification of components (see the **detail block: Unique identification of software**) it can be necessary in certain cases to determine the identity of components (*entity resolution*). There are support tools for this purpose too but it can be a complex process that also requires manual work.
- Depending on the depth of the SBOM, it may be necessary to search for dependencies on dependencies (*transitive dependency resolution*). The number of dependencies that have to be worked through can grow rapidly, so the use of tools for this task is recommended.

Close detail block

Detail block:
Unique identification for software

To enable the effective use of SBOMs, it is essential that software components are identified in the same way, everywhere. Only then can a software component from an SBOM be automatically linked to known vulnerabilities, licences and other relevant information. If this does not happen, *false positives* or *false negatives* can arise. In the case of a *false positive*, a match is made between information that is not actually related. If this happens when searching for vulnerabilities, a component can wrongly be identified as vulnerable. In the case of a *false negative*, actually no match is made where it should be. This means that a component may appear to not be vulnerable, while in fact it is.
Unfortunately, at present, too many different ways of identifying software components are in use. The most commonly used methods are:

- Coordinates
- Package URL (PURL)
- Common Platform Enumeration (CPE)
- Software Identification (SWID) tagging
- Cryptographic hash functions (SHA-1, SHA-2, SHA-3, BLAKE2b, BLAKE3)

The use of **coordinates** coordinates is described as: *group, name* and *version*. An example is "group": "org.example", "name": "sample-library", "version": "1.0.0.

PURL was the result of an initiative to introduce a standard method of identifying software packages, independently of the context of a specific package manager. It makes use of a simple syntax in which a number of components are combined to form a URL: scheme:type/namespace/name@version?qualifiers#subpath. This results in identifiers such as: pkg:maven/org.apache.xmlgraphics/batik-anim@1.9.1?packaging=sources.

CPE is a commonly used standard, in particular in combination with vulnerability and incident management. However, the process surrounding the setting of a CPE is unfortunately vulnerable to human error. In addition, the granularity of the CPE approach is too limited in certain situations to determine whether a software component actually contains a vulnerability. For that reason, the SBOM Forum recommends switching to the use of PURL [17].

1 2 →

Close detail block

The **Software Identification** (SWID) Tag is an ISO-standaard [16↗] which was already defined in 2012 as a means of maintaining an overview of the software installed within an organisation. In practice, in the context of SBOM, SWID Tags are primarily used as a means of identifying software components within the CyclonDX or SPDX format, which is explained in the section on *SBOM Standards*.

The use of **hashes** is a well-known means of verifying that two pieces of code are identical. In practice, this proves difficult to use because different hash functions are used or because it is unclear precisely which components of a package have been included in the hash. For example, is or is not the SBOM itself part of the hash if delivered as part of the package?

In addition to the use of different standards, the names used by suppliers for software are also subject to changes, for example as a result of mergers and acquisitions between the organisations, or as a result of new project forks.

The conclusion is that there is still a long way to go before a uniform global approach will be introduced for identifying software components. Until that time, the NTIA recommends applying the following principles [14↗]:

- If a uniform method already exists for identifying a component, follow this method. Examples are names from package managers that use unique identifiers or names from suppliers that allocate clear identifiers to their software
- If no uniform method yet exists, choose an existing, commonly used standard to describe a component.

Producing, Managing & Sharing SBOMs

03.

Back to starting figure

The management of SBOMs within an organisation involves two aspects: on the one hand ensuring that SBOMs are up to date and on the other managing the total collection of SBOMs. Both aspects are briefly explained below.

Throughout their *lifecycle*, software applications often undergo changes, such as *bug fixes, security patches, new features*, etc. [5↗]. Every time the software is altered, the software supplier must also supply a new SBOM that includes the changes. In theory, the software configuration carried out by the software customer can also be added to the SBOM. In this way, it is possible to distinguish between different configurations which may also have different runtime dependencies. In practice, however, as yet little is known about how this can be used. The advice is not to set up a separate process for managing the SBOMs over time, but to link this process to already existing processes for software configuration management.

To manage the total overview of SBOMs, the same aspects are relevant as those that apply to the management of other large volumes of data: storage capacity, indexing, backup systems, etc. A specific point for attention for SBOM storage is security. It is after all essential that the organisation can trust the SBOMs, and for that reason the risk of SBOM sabotage by malicious parties must be minimised [8↗].

Back to starting figure

Step 3:
Organise the distribution of SBOMs to third parties

SBOMs can be distributed to other organisations in a number of different ways. A number of examples [9↗]:

- The SBOM is downloaded via the website/API of the supplier. The address appears in a pre-agreed location in the software.
- The SBOM is sent by email or other communication channel.
- The SBOM is supplied as part of the software package or the embedded system.

When making choices about the structure of the distribution process, the following factors must be taken into account:

- **Offline availability**: for embedded systems, among others, it can be important to have an option for inspecting the SBOM on the system, at all times, even without online integration. In this situation, the SBOM must be an integral part of the software package.
- **Possibility of automation**: a major benefit of SBOM is the possibility of automation of a number of processes. To be able to benefit from this advantage, the distribution/reception of SBOMs will also have to be automated. This argues in favour of solutions such as APIs, rather than email or other information communication channels..
- **Scalability**: this point is an extension of the previous two points. The expectations is that the number of SBOMs managed by an organisation will quickly grow considerably, for example because of different customers, different versions, etc. This could be a reason to consider setting up an API with push/subscribe options, in which SBOMs are automatically shared.

At present, there are still few best practices relating to the sharing of SBOMs. The CISA has organised an SBOM Workstream relating to SBOM *'Sharing & Exchanging'* in which they are working to develop recommendations, together with the community.

Go to step 4 →

Producing, Managing & Sharing SBOMs

03.

Back to starting figure

Step 4:
Evaluate and improve

The use of the SBOM and the accompanying tooling is still a relatively new phenomenon, and will undergo numerous developments over the coming period. It is therefore advisable to follow a phased approach starting with the low-hanging fruit and which leaves space for incorporating new best practices or tooling. To make this possible, a continuous improvement approach could be used such as the Plan-Do-Check-Act cycle. In this process, it is important to identify concrete/measurable targets in the plan phase, and to reach clear agreements for the do phase. The advice is to start small and to gradually discover which processes operate smoothly and where further information or connections are missing. In the check phase, also involve the relevant parties from outside your own organisation, such as software suppliers and peer organisations. This makes it possible to reach joint decisions on alternative approaches where necessary, and the relevant processes and toolchains will be established in an iterative process.

Deploying SBOMs for Vulnerability Management

Deploying SBOMs

04.

Deploying SBOMs for Vulnerability Management

This chapter describes how SBOMs can be deployed to reinforce the vulnerability management process. To make optimum use of the benefits of SBOM, it is heavily recommended to also immediately integrate the use of the Vulnerability Exploitability eXchange (VEX) security advisory. For that reason, this chapter also describes how the combination of SBOM and VEX can be employed to ensure a more effective vulnerability management process.

In this guide, vulnerability management is defined as a continuous, proactive and risk-driven process used by organisations to secure their systems against cyber threats. In this process, potential vulnerabilities are identified and interpreted, at which point prioritised vulnerabilities are tackled on the basis of a risk assessment. Because it is a continuous process, it is often also referred to as the vulnerability management cycle. The precise way in which this cycle is structured will differ from organisation to organisation, but in broad terms, three steps can be identified [10↗]:

- Know which assets the organisation owns and when vulnerabilities in respect of those assets are discovered.
- Create an insight into the impact of vulnerabilities and which mitigating measures need to be implemented.
- Implement the chosen mitigating measures.

The idea behind using SBOM for vulnerability management is that SBOMs can accelerate the process of searching for vulnerable software components and their dependencies. This is because SBOMs provide an overview of which software components are used and where they are located. As a consequence, security specialists can use their already scarce time to focus on prioritising and mitigating the vulnerabilities. One point that does deserve attention in this connection is that a better total overview of all software components combined with an ever growing number of known vulnerabilities will result in the identification of more and more potential vulnerabilities in step 1 of the vulnerability management cycle. The ability to rapidly estimate which vulnerabilities deserve further investigation will itself become increasingly important as a consequence. VEX is expected to play an important role in this process. The following sections first provide an explanation of what VEX is, before considering how SBOM and VEX can be used together in the vulnerability management process.

What is VEX? →

What is VEX?

VEX is a type of security advisory according to which software suppliers can indicate the actual impact they expect a specific vulnerability to have on a product [11↗]. A VEX document is machine-readable and contains at least the following information [12↗]:

- The metadata, to identify the document and the author.
- The product details, such as the identifier and the software version number.
- Details about the vulnerability, including an identifier and an explanation.
- The VEX status for the product.

The VEX status is the field which indicates whether the software is vulnerable. The supplier can choose from the following options [12↗]:

- not affected,
- affected,
- fixed, or
- under investigation.

The next section describes how to deal with the various options.

As well as warning consumers about serious vulnerabilities, VEX statuses can in fact also be used to reassure consumers that certain vulnerabilities do not apply to them. Sometimes a software application does contain a vulnerability component but nevertheless does not represent a risk because the vulnerability is compensated for in some other way or because the affected piece of code is not called up. A supplier is able to make this analysis and subsequently inform its entire customer base.

Integrating SBOM and VEX →

Integrating SBOM and VEX in the Vulnerability Management Cycle

Having explained all the building blocks, this section describes how SBOM and VEX can be combined in the vulnerability management process. The description refers back to the steps of the vulnerability management cycle described above.

Step 1: Know which assets the organisation owns and when vulnerabilities against those assets are discovered.
By setting up the production and management of SBOMs as described in the previous chapter, the organisation has access to a continuous, up-to-date overview of the software components used. As soon as a new vulnerability is made known, it can be entered into a vulnerability management tool at which point the tool automatically indicates which software packages contain the vulnerable component. Depending on the tooling chosen, different options will be made available for linking vulnerabilities to software components from an SBOM. A number of options:

- The tool has a direct link to a vulnerability database such as the NIST National Vulnerability Database or MITREs Common Vulnerabilities and Exposures database. As soon as a new vulnerability is added, the tool scans the SBOM database to determine which components are vulnerable.
- The organisation receives information about a new vulnera-bility via a non machine-readable format such as an email, website or telephone call. A security analyst then manually enters this information in the tool, at which point the tool scans the SBOM database for vulnerable components.
- The organisation receives information about a new vulnerability via a machine-readable format such as a VEX document. These documents can be directly entered by the tool. Work is currently underway on a standard that makes this possible: the Common Security Advisory Framework (CSAF). For more information see the **detail block**

In particular at this stage, the SBOM will save a great deal of time. As soon as the information about a vulnerability has been loaded into the vulnerability management tool, the security analyst will be able to see within just a few seconds in which application within the organisation the vulnerability is present.

Remember that there will always be software components within an organisation that are not or not fully covered by an SBOM

Deploying SBOMs

04.

Back to starting figure

Close detail block

Detail block:
CSAF

At present, information about vulnerabilities is shared in many different ways. For example via mailing lists or blogpost, or in urgent cases via a telephone call from a supplier or the NCSC. The Common Security Advisory Framework (CSAF) has been developed to standardise the form of security advisories and to make them machine-readable. This makes it possible to distribute information about new vulnerabilities more rapidly, and to process that information automatically [20↗]. CSAF is managed by the standardisation group OASIS Open and is actively promoted by NTIA (US) and BSI (DE). CSAF is the replacement for the Common Vulnerability Reporting Framework (CVRF). It supports a number of different profiles for various types of advisories. One of the profiles supported by CSAF is VEX. The CSAF website offers several videos explaining the concepts behind CSAF, as well as the combination of CSAF and SBOM.

vulnerable component. Depending on the tooling chosen, different options will be made available for linking vulnerabilities to software components from an SBOM. A number of options:

- The tool has a direct link to a vulnerability database such as the NIST National Vulnerability Database or MITREs Common Vulnerabilities and Exposures database. As soon as a new

of time. As soon as the information about a vulnerability has been loaded into the vulnerability management tool, the security analyst will be able to see within just a few seconds in which application within the organisation the vulnerability is present.

Remember that there will always be software components within an organisation that are not or not fully covered by an SBOM

1 2 3 →

(open-source software without SBOM, for example). For software components, this means that a risk assessment must be carried out in advance of the extent to which they are sufficiently covered by SBOM data. It is also important at that moment to determine how to deal with critical vulnerabilities that could apply to the component (can the application in which the component is used be temporarily shut down, while determining whether the component is vulnerable).

Step 2: Create an insight into the impact of vulnerabilities and which mitigating measures need to be implemented.

At this stage, the added value of VEX becomes clear. After step 1 has been implemented, the analyst will be faced with a list of a possibly large number of potentially vulnerable applications. That is why it is so important to reach clear agreements in advance with software suppliers about the supply of VEX documents. Ideally, in the event of serious vulnerabilities, the supplier will automatically send a VEX document to its customers. According to alternative agreements, the organisation can ask the supplier for a VEX status relating to a specific vulnerability. As soon as the VEX is received, it can be automatically read by the vulnerability management. The status allocated in the document determines the next step:

- **Not affected**: the vulnerability has no impact on the product and no further action is required.
- **Fixed**: this status is often issued in combination with an *affected status*. It can for example mean that the current product version is not influenced but previous versions are. In principle, this status means that no action is currently required. However, if the previous (vulnerable) version was also used by the organisation, and the vulnerability relates to a critical system, it may still be essential to carry out a more in-depth investigation to be certain that malicious parties have not already made use of the vulnerability while the old version was still in use.
- **Under investigation**: this status is issued until the supplier has sufficient information to update the status. The agreements with the supplier must also include an agreement on how to deal with this status. It is recommended that you agree that the customer will not seek to contact the supplier, to allow the supplier to carry out the investigation, rather than having to respond to concerned customers.
- **Affected**: the product is vulnerable to the new threat. In these cases, the supplier must specify which steps the customer needs to take in order to minimise the risk of the vulnerability,

Deploying SBOMs

04.

for example updating the product to a safe version. The customer must then implement the mitigating measures via its own vulnerability management process (this then is step 3).

If no VEX status is supplied, it will be up to the analyst to carry out the risk assessment to determine whether a software package with a vulnerable component is actually vulnerable.

As soon as it becomes clear whether a software product is or has been vulnerable, the security analyst must determine the actual impact of a vulnerable software product for the organisation, which mitigating measures are possible for the organisation, and their potential impact on the organisation. The way in which these considerations should be made is the same as how this process should be carried out without SBOM and VEX, and as such is beyond the scope of this guide. Specifically for vulnerabilities in commonly used components (such as with the Log4J incident), SBOM is able to offer a greater insight into the total picture of the full range of software products that are influenced, thereby making it possible to arrive at a more balanced consideration of the total package of mitigating measures and their impact.

Step 3: Implement the chosen mitigating measures.
In this step, the chosen mitigating measures are actually implemented. This often involves a preparation and a test phase, as well as a rollout phase, after which a check must be carried out to determine whether the measures have indeed had the desired effect. The actual implementation of this step is not directly influenced by the use of SBOM and VEX.

If the underlying processes and tooling are well matched, the combined deployment of VEX and SBOM can help automate a time-consuming element of the vulnerability management process. This matching or harmonisation process is no trivial task and many aspects relating to interoperability and standardisation still need to be further elaborated. A determination will have to be made for each individual organisation about how the processes are set up and which combination of support tools best suits them. Nevertheless, all relevant parties strongly recommend making a start on the use of SBOM for vulnerability management. This tool requires a phased approach, with space for evaluation and adjustments.

Afterword

Back to starting figure

SBOM is an important building block in enhancing the transparency of the software supply chain and reinforcing security. The expectation is that in the future, software will always have to be supplied with an SBOM. At present there is still much discussion about what SBOMs and the related processes should look like.
This offers room for experimentation and learning. The final recommendation at the end of this guide is therefore to keep a close eye on the initiatives that could be relevant to your organisation. A number of initiatives currently underway are:

- The approval and introduction of the EU Cyber Resilience Act; an update is expected in June 2023 [13↗]
- CISA SBOM Workstreams: The American Cybersecurity & Infrastructure Security Agency (the American NCSC) has established four working groups to consider the various aspects of SBOM:
  - The Cloud & Online Applications working group will regulate the use of SBOMs in cloud and SaaS applications.
  - The On-ramps & Adoption working group is attempting to broaden awareness of SBOM, as a means of helping organisations that still have little knowledge of this subject to get on track.
  - The Sharing & Exchanging working group is investigating how to deal with the complex issue of the requirements that must be satisfied to make it possible to share SBOMs between organisations.
  - The Tooling & Implementation working group is concentrating on the possibilities and challenges of automating SBOM processes using tooling.
- ISO/IEC 27036-3 - Guidelines for information and communication technology supply chain security: This ISO standard is currently being developed. Part 3 above all focuses on guidelines for hardware, software and supply chain security.
- Dependency track: Dependency track is a tool capable of producing and analysing SBOMs. The tool was developed as an open-source project by the OWASP community and is open to active further development by participants in that community.
- Global Platform SBOM Task Force: Global Platform develops and publishes standards for secure chip technology. They have launched a working group to analyse the influence of SBOMs in the secure chip world. This working group could be of real interest to secure chip suppliers because secure chips are used in a so many different sectors (for example the car industry, financial sector, energy sector, etc.).

Many of these initiatives are taking place at international and cross-sectoral level. It could therefore also be valuable to organise similar discussions with peer organisations within your own sector, in order to consider more sector-specific aspects.

References →

Back to starting figure

References

1. **Cyber Safety Review Board**. Review of the December 2021 Log4j Event. CISA. [Online] 11 July 2022.

2. **Riel, Bart van, Kuijpers, Sanne en Koning, Roeland de**. *Using the Software Bill of Materials for Enhancing Cybersecurity*. sl : Capgemini Invent, 2021.

3. **NTIA Multistakeholder Process on Software Component Transparency Use Cases and State of Practice Working Group**. Roles and Benefits for SBOM Across the Supply Chain. *NTIA*. [Online] 8 November 2019. [Quote dated: 12 February 2023.]

4. **The United States Department of Commerce**. The Minimum Elements For a Software Bill of Materials (SBOM). *The National Telecommunications and Information Administration*. [Online] 12 july 2021.

5. **NTIA**. Survey of Existing SBOM Formats and Standards. *NTIA*. [Online] 2021. [Quote dated: 20 January 2023.]

6. **NTIA Formats and Tooling Working Group**. Software Suppliers Playbook: SBOM Production and Provision. *NTIA*. [Online] 17 November 2021. [Quote dated: 13 February 2023.]

7. **NTIA Formats and Tooling Working Group**. Software Consumers Playbook: SBOM Acquisition, Management, and Use. *NTIA*. [Online] 17 November 2021. [Quote dated: 13 February 2023.]

8. **Muro, Iradier Alvaro**. SBOMs 101: What You Need to Know. *DevOps*. [Online] 19 July 2022. [Quote dated: 14 March 2023.]

9. **NTIA**. Sharing and Exchanging SBOMs. *NTIA*. [Online] 10 February 2021.

10. **Souppaya, Murugiah en Scarfone, Karen**. Guide to Enterprise Patch Management Planning: Preventive Maintenance for Technology. *NIST*. [Online] April 2022. [Quote dated: 8 March 2023.]

11. **NTIA**. Vulnerability-Exploitability eXchange (VEX) – An Overview. *NTIA*. [Online] 27 September 2021.

12. **VEX Working Group**. Vulnerability Exploitability eXchange (VEX) - Use Cases. *CISA*. [Online] April 2022. [Quote dated: 6 March 2023.]

13. **Car, Polona**. Horizontal cybersecurity requirements for products with digital elements. *Legislative Train Schedule*. [Online] European Parliament, 20 May 2023. [Quote dated: 4 June 2023.]

Back to starting figure

References

14. **NTIA Multistakeholder Process on Software Component Transparency Framing Working Group**. Software Identification Challenges and Guidance. *NTIA*. [Online] 30 March 2021. [Quote dated: 12 February 2023.]

15. **Synopsys**. What is the impact of the New EU Cyber Resilience Act on the Software Supply Chain SBOM? [Webinar] sl : AppSecNL & Synopsys, 24 November 2022.

16. **International Organization for Standardization**. ISO/IEC 19770-2:2015. *ISO*. [Online] March 2017. [Quote dated: 12 February 2023.]

17. **The SBOM Forum**. OWASP. A Proposal to Operationalize Component Identification for Vulnerability Management. [Online] 13 September 2022. [Quote dated: 12 February 2023.]

18. **Europese Commissie.** Regulation of the European Parliament and of the Council on Horizontal Cybersecurity requirements for products with digital elements and amending Regulation (EU) 2019/1020. *European Commission*. [Online] 15 September 2022. [Quote dated: 4 June 2023.]

19. **Biden jr., Joseph**. Executive Order on Improving the Nation's Cybersecurity. *The White House*. [Online] 12 May 2021.

20. **Friedman, Allan en Schmidt, Thomas**. Your Software IS/NOT Vulnerable: CSAF, VEX and the Future of Advisories. *YouTube*. [Online] 6 December 2021.

Back to starting figure

Publication details

This guide was compiled as part of the multiyear collaboration agreement between the NCSC and TNO on reinforcing supply chain management.

Auteurs:

Gwen Jansen-Ferdinandus
Niels Brink
Silke Mergler
Andre Smulders
Peter-Paul Meiler

The authors would like to express their gratitude to the following experts and organisations for their contribution and feedback during the creation of this guide: Bart de Wijs from ABB, Allan Friedman from CISA, Philips, RDI, Siemens and UWV (Employee Insurance Agency).

Nationaal Cyber Security Centrum
Ministerie van Justitie en Veiligheid

TNO innovation for life