Capgemini

Nationaal Cyber Security Centrum
Ministerie van Veiligheid en Justitie

TLP:WHITE

Using the Software Bill of Materials for Enhancing Cybersecurity

Capgemini Invent

Bart van Riel, Sanne Kuijpers, Roeland de Koning

January 2021

Table of Contents

For more information, please contact:
Roeland de Koning – roeland.de.koning@capgemini.com
Bart van Riel – bart.van.riel@capgemini.com
Sanne Kuijpers – sanne.kuijpers@capgemini.com

1. Management Summary

Modern software systems involve increasingly complex and dynamic supply chains. Lack of systemic visibility into the composition and functionality of these systems contributes substantially to cybersecurity risk as well as the costs of development, procurement, and maintenance. The increasing focus on security aspects of complex IT landscapes and supply chain integrity drives the adoption of a standardized information carrier describing the internals and origins of software components to achieve software transparency.
The Software Bill of Materials (SBoM) is an electronic document or machine readable file describing the parts that a piece of software consists of. This transparency allows the using organization to be more aware of vulnerabilities in underlying software components and to better assess IT landscape impact of those vulnerabilities. Additionally, an SBoM enhances risk management for choosers and operators of IT resources.

The Software Bill of Materials is clearly gaining tracktion within the IT security world. The National Telecommunications and Information Administration (from hereon: NTIA), part of the United States Department of Commerce, has taken up the task of organizing a multi-stakeholder approach to come to an agreed usage and format of an SBoM by establishing various working groups which engage stakeholders and collect insights. ENISA mentions the SBoM as a device for increasing security within the context of IoT and medical devices. However, there does not yet seem to be a great deal of standardization in the actual usage of the SBoM at this point in time. Accepted data standards and tool support seem to be limited.

One of the areas where software transparency is very common is in the world of medical device certification. A fully traceable list of software parts is a mandatory requirement to obtain a CE certification, very similar to the content of an SBoM. Security aspects are also included: CE standards require by that medical device producers actively inform their medical institution consumers of vulnerabilities.

Modern IT landscapes show an increasing variety in application solutions, which leads to increasing complexity in controlling them. However, you can only control what you know. Any additional information about the inherent structure of your software solutions can lead to greater control and increased security posture. An SBoM provides value to increased security posture. That value is approached through four different perspectives: 1) Producing, 2) Choosing, 3) Operating and 4) SecDevOps.

Perspective: Producing

An SBoM should be made available for any released version of a software product, as part of product documentation. It is recommended to include the updates to SBoM information into the Definition of Done (or the organizations development approach equivalent of the DoD) and to include the availability of an SBoM as a test case, and to include SBoM information automated updates to be a standard part of each build cycle in the CI/CD pipeline.

Figure 1 Usage of the SBoM in software development

One of the more unfortunate results of automatically built library repositories, is that many different versions of the same library get imported into your component repository. All those versions will have their own security vulnerabilites so one of the ways to improve code security is to reduce the variety of related component versions. The software transparency that an SBoM delivers, provides knowledge of the level of technical variation underlying your software's implementation. Such knowledge also allows proactive assessment of vulnerabilities in your code, inherited through component dependencies.

Perspective: Choosing

The process of choosing a software component usually employs many steps of information gathering and selection. An important consideration should always be how the software component may impact the security posture of the organisation by having security vulnerabilities. An SBoM can contribute to the security posture impact assessment by allowing preliminary validation of such vulnerabilities. Preliminary validation is achieved by ad-hoc matching of a software components' SBoM against vulnerability listings in known CVE repositories.
Effective ad-hoc vulnerability matching must be supported by digital tooling which provides visual ways of flagging potential vulnerabilities. This can even be done to verify the trust level of source code providers underlying the software components embedded integrations.
The identified vulnerabilities can then be taken into account to form a balanced view of the security posture impact, before promoting the chosen software component to an operational production environment.

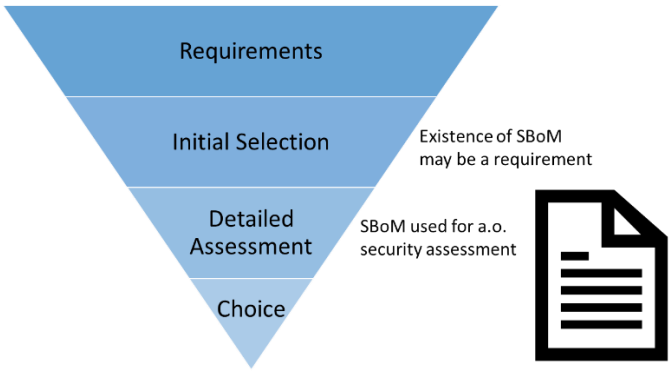


Figure 2 Usage of the SBoM when choosing IT components

Perspective: Operating

An important aspect of Secure Operations is the ability to continuously assess the IT landscape for potentially vulnerable software components. The organisation can mitigate any risks on their own accord.

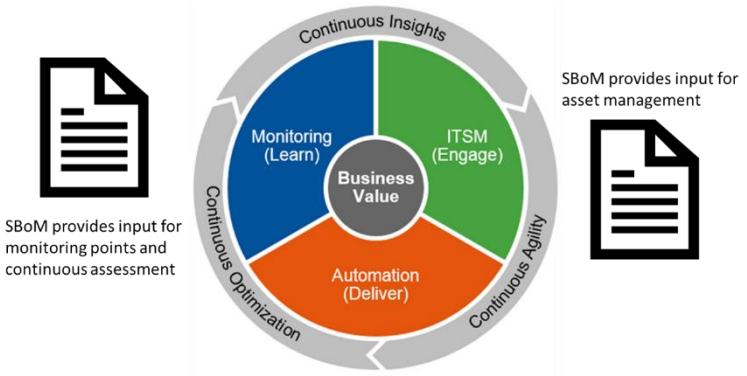


Figure 3 Usage of the SBoM in Operations

Continuous assessment requires automated matching of SBoM information of software components against CVE repositories. The information in an SBoM enables such continuous assessment by providing part of the data as input for such automation. Automated matching requires standardized and correlatable data elements of software components SBoMs and CVE repositories, contained in agreed data formats and with validatable completeness, and containing compatible data values.

Perspective: SecDevOps

SecDevOps essentially combines the Perspectives on Producing and Operation. DEVops, where it's all about developing the software, is linked to the Producing Perspective. Where the devOPS is linked to the Operation, because it is about bringing the (new) software into the operation through release, deploy, operate and monitor.

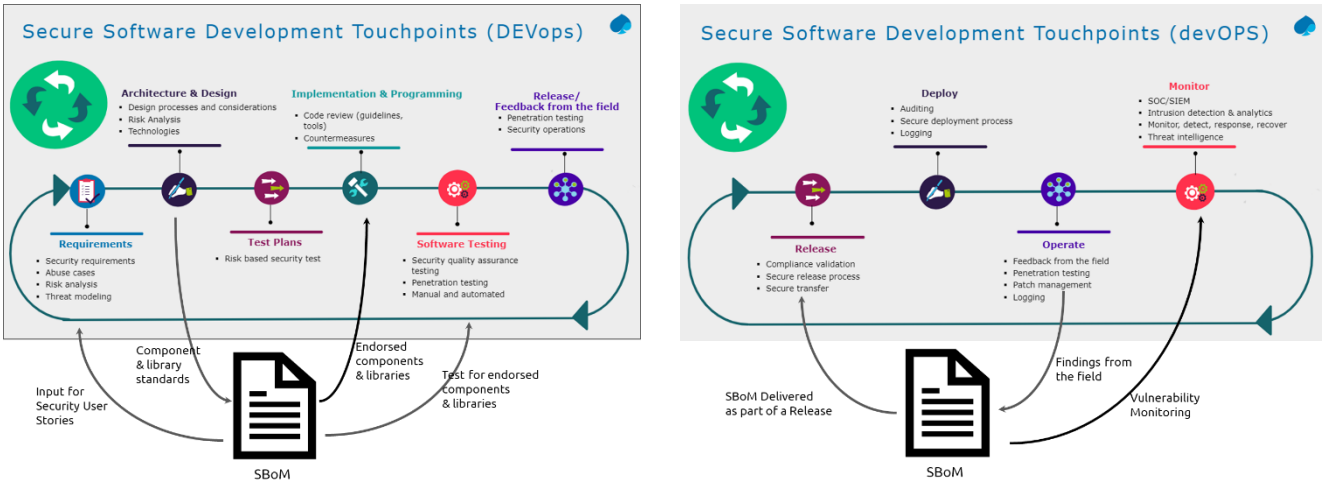


Figure 4 Usage points of the SBoM in DEVops and devOPS

During development, the SBoM can describe specific libraries (or the libraries that you can't use!), charactersets, etc. An SBoM would be made available for any released version of a software product, as part of product documentation. It is important that parts of the SBOM information will be tested every sprint as well. During operations, the SBOM should be monitored and updated on the findings

discovered. These could be based on feedback from the field, but they can also be discovered through monitoring.

Online and offline availability of the SBoM

The optimal way of making an SBoM available depends on the usage scenario, but in general the SBOM should contain up-to-date information so any mechanism that supports keeping the SBOM up-to-date continuously has preference.

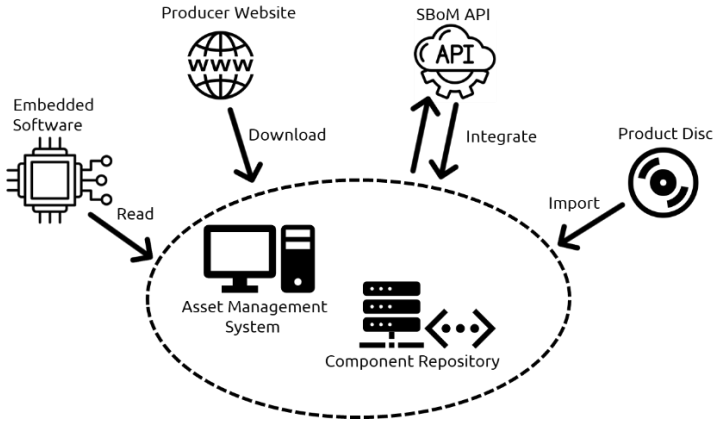


Figure 5 An overview of the methods to make an SBoM available

An SBOM can be made available electronically for example via download to be obtained whenever the information is needed. Thinking in a more integrated fashion, the SBOM data could be made available via an exposed API to be integrated into for example build systems or asset management systems. For embedded software in devices which are deployed in a distributed manner, it may make more sense to package the SBoM with the embedded software on the device.

Fit for Purpose SBoM Information

One of the key success factors for enhancing security is to have fit for purpose information in the SBOM. *Level of Granularity* is driven by the intended 'depth of control' of your software chain. Deeper control requires more granular information. *Information Content Attributes* must enable the security position you want to achieve. Different security controls require different types of information. *Data standards* are a requirement for automated correlation (e.g. with a CvE database, or a CMDB). Regarding data values, there is at least a need for an authorative source for Component Name, Version and Supplier. *Data Formats* in an SBoM should be self-describing, and structured, using a schema syntax and a name-space mechanism. The structure should also be self-validating by clearly marking required data attributes.

CycloneDX as preferred SBoM Data Format

CycloneDX (https://cyclonedx.org/) can combine the unambiguous software component identity and their relationships to other software components. CycloneDX contains a vulnerability schema extension to hold URL references to CVE entries.

2. Introduction

2.1 Context of the report

Modern software systems involve increasingly complex and dynamic supply chains. Lack of systemic visibility into the composition and functionality of these systems contributes substantially to cybersecurity risk as well as the costs of development, procurement, and maintenance. In our increasingly interconnected world, risk and cost impact not only individuals and organizations directly but also collective goods like public safety and national security.[1]
The increasing focus on security aspects of complex IT landscapes and supply chain integrity drives the adoption of a standardized information carrier describing the internals and origins of software components to achieve software transparency. The Software Bill of Materials (SBoM) is an electronic document or machine readable file discribing the parts that a piece of software consists of. The SBoM contains a list of one or more identified components, information about those components (e.g. version numbers), and the supply chain relationships between those components (called 'pedigree' or 'provenance').
The information contained in an SBoM can play a role in increasing an organisation's cyber security position by providing transparency of the composition of Software-IT resources (such as applications and embedded software) from software producers. This transparency allows the using organization to be more aware of vulnerabilities in underlying software components and to better assess IT landscape impact of those vulnerabilities. Additionally, an SBoM enhances risk management for choosers and operators of IT resources.

The NCSC has requested an exploratory study of the current position of the SBoM, the landscape in which it can be deployed and the additional IT security that the SBoM can help to provide. This report contains the findings, conclusions and recommendations from that study. It is good to note that the study is focused on the supply chain and software security in a B2B context.

2.2 Structure of the Report

Chapter 3 summarizes experiences with the Software Bill of Materials thus far in both the US and within Europe.
Chapter 4 discusses the position of the SBOM within the supply chain from four different perspectives: Producing, Choosing, Operating and SecDevOps. Chapter 4 also provides suggestions on further integration off the SBoM into those perspectives to achieve a higher level of cyber security in IT landscapes.
Chapter 5 discusses key success factors for enabling vulnerability detection with the use of the SBoM, such as the required information contained within an SBoM, possible scenarios to make an SBoM available online and offline, as well as use cases and tooling requirements to make effective use of an SBoM to achieve a higher level of security.
Finally chapter 6 gives a short overview of future expectations for furthering the usage of the Software Bill of Materials.

2.3 Methodology

The information in this report was compiled in various ways. We started with extensive desk research to gain an understanding on the current views on SBoM usage and the potential for enhancing cybersecurity. Several chapters in this document contain combined insights from available source materials, which are listed in the bibliography in 7.3.

[1] https://www.ntia.gov/files/ntia/publications/framingsbom_20191112.pdf

This desk research was iteratively combined with insights from internal and external experts, collected through interviews. The collected insights were reflected upon together with field experts in a workshop. The reflections and comments from the workshop were fed into the report. We have been in close contact with NCSC throughout the process and were fortunate to have access to their expert contacts.

3. State of the SBoM

This chapter explores the current state of adoption and usage of the concept of the Software Bill of Materials in the US and Europe.

3.1 US Experiences

3.1.1 Cyber Supply Chain Management and Transparency Act of 2014

On December 4th 2014 Representative Edward Royce introduced a Bill in the US House of Congress, with the proposed Act title "Cyber Supply Chain Management and Transparency Act of 2014", detailing measures for increased software transparency with the explicit purpose of enhancing security (https://www.congress.gov/bill/113th-congress/house-bill/5793). Although the Bill has never been voted into an Act, it got the ball rolling on a discussion on the merits of regulated (and standardized) aproaches to software transparency. It more or less formalized the concept of the Software Bill of Materials (from hereon: SBoM).

3.1.2 NTIA (National Telecommunications and Information Administration)

The National Telecommunications and Information Administration (from hereon: NTIA), part of the United States Department of Commerce, has taken up the task of organizing a multi-stakeholder approach to come to an agreed usage and format of an SBoM by establishing various working groups which engage stakeholders and collect insights. The following NTIA working groups are currently in operation:

- Framing Working group: Establishing introductory SBOM documentation for framing the concept of an SBoM to interested parties.
- Awareness and Adoption Working Group: Creating an agreed overview of use cases for achieving the benefits of an SBOM.
- Formats & Tooling Working Group: Performing an ongoing survey of SBOM related tooling, data formats and standards.
- Healthcare Proof of Concept Working Group: Facilitating and reporting on an ongoing PoC on SBoM usage in the healthcare sector (US).

(https://www.ntia.gov/SBOM and https://www.ntia.gov/SoftwareTransparency)

3.1.3 CISQ and OMG

The CISQ Working Group: "Tool-to-Tool Software Bill of Materials Exchange" is a joint working group of CISQ (Consortium for Information & Software Quality) and the OMG (Object Management Group) with the objective of defining an exchangeable tool-to-tool bill of materials metamodel for software (SBOMs) and other items needing BOMs (https://www.it-cisq.org/software-bill-of-materials/)

3.1.4 MITRE Corporation and NIST

MITRE Corporation and NIST have published an approach for using the SBoM to record and validate software "Provenance" (Chain of Custody, 'the path of the software between organizations') and Pedigree (Lineage, 'record of origin steps in the software supply chain').
(https://csrc.nist.gov/CSRC/media/Projects/cyber-supply-chain-risk-management/documents/SSCA/Spring_2019/8MayAM2.3_Software_Bill_of_Materials_Robert_Martin_05_08_19_clean.pdf)

3.2 European Experiences Regarding Software Transparency and Traceability

3.2.1 Medical Devices in Hospitals

One of the areas where software transparency it's very common is in the world of medical device certification. For example, the CE[2] Marking for Medical[3] Devices (a certification scheme mandatory for any seller of medical devices in Europe) requires that the device producer keeps fully traceable records of the composite parts off the medical device down to the level of raw material sourcing. A fully traceable list of software parts is a mandatory requirement to obtain this CE certification. Such a full traceability requirement regarding software transparency requires a form of records keeping which very similar to the content of an SBoM.
The requirements extend beyond software and into manufacturing – the producer is also required to maintain detailed descriptions of procedures surrounding the implementation of software and its support. Security aspects are also included: within the healthcare sector it is common and required by CE Marking standards that medical device producers actively inform their medical institution consumers of vulnerabilities.
There is room for improvement though. The vulnerability advisories from the producers are not provided in a standard data format which is digestible by automated asset management systems. Within the healthcare sector, there is a need for reducing manual work load of vulnerability assessments through advanced automation.

3.2.2 Position of the SBoM in Medical Device Software Manufacturing

Supply chain security is a crucial part of medical device manufacturing, made mandatory by certifications. Supply chain extends from the product to support and maintenance onsite at the devicer users
A Software Bill of Materials, representing the full internal trace of embedded software components, is used in varying levels of content granularity. Within the software development process, the granularity level is very high allow for extensive software risk mitigation. Towards the end user, the SBoM level of granularity is lower.
There is extensive discussion on the desired level of detail in an SBoM. Intellectual Property aspects actually do not really play a role here, however the limits to effective use do. Effective use of SBoM details is highly dependent on the IT management maturity of the organisation using the software. IT management maturity varies greatly within the medical care industry. To facilitate the absorption of security (and privacy) aspects of medical devices for medical care organisations, the MDS2 [4] (Manufacturer Disclosure Statement for Medical Device Security) was introduced by the medical device manufacturer community. The existence of an SBoM a specific part of the MDS2 content.

3.2.3 Governmental and Resarch bodies

ENISA mentions the SBoM as a device for increasing security in two reports. In the report "Guidelines For Securing The Internet Of Things" [5], ENISA recommends to use the existence of an SBoM for IoT devices and suggests the tool DependencyTrack to identify any underlying software dependencies and generate the SBoM. In the report "Procurement Guidelines For Cybersecurity Inhospitals" [6], ENISA

[2] "Conformité Européenne", https://ec.europa.eu/growth/single-market/ce-marking_en
[3] https://www.ema.europa.eu/en/human-regulatory/overview/medical-devices
[4] https://www.nema.org/Standards/view/Manufacturer-Disclosure-Statement-for-Medical-Device-Security and https://www.nen.nl/media/wysiwyg/NEN_WhitePaper_GebruikMDS2_def.pdf
[5] https://www.enisa.europa.eu/publications/guidelines-for-securing-the-internet-of-things

[6] https://www.enisa.europa.eu/publications/good-practices-for-the-security-of-healthcare-services

encourages to consider including a requirement for a BOM (Bill of Materials) for hardware and software when choosing medical devices. The two reports do not provide details on how to integrate the SBoM in existing IT processes, or any associated standards.

3.3 Summary: State of the SBoM

The Software Bill of Materials is clearly gaining tracktion within the IT security world, both explicitly name so as well as the concept of software transparency. The existence of an SBoM is generally considered as an indicator of IT product quality. Also, the SBoM is considered to hold valuable information for managing IT security. The ideal balance between SBoM detail level versus practical usability requires ungoing discussion. Unfortunately, there does not yet seem to be a great deal of standardization in the actual usage of the SBoM at this point in time. Accepted data standards and tool support seem to be limited. Also, research has shown that applicable best practices in using the SBoM for security enhancement are limited as for now.

4. Position of the SBoM in the IT Supply Chain

4.1 Software Transparency in the IT Supply Chain

Modern IT landscapes show an increasing variety in application solutions, which leads to increasing complexity in controlling them. Additionally, apart from The IT landscape of an organization, the greater variation in IT solutions leads to a higher level of complexity of the supply chains delivering those solutions. At the same time, requirements on governance and security posture induce a greater desire for control of both the IT landscape and the IT supply chain.
However, you can only control what you know. Any additional information about the inherent structure of your software solutions can lead to greater control and increased security posture. A Software Bill of Materials (SBoM) provides such additional information.
This chapter examines the value that the SBoM brings to increased security posture. That value is approached through four different perspectives:

1. The *Producing* perspective, which is the perspective for the creation and continuous maintenance of software;
2. The *Choosing* perspective, which is the perspective for selecting software still solutions to use within a night landscape;
3. The *Operating* perspective, which is the perspective concerned with ensuring that the landscape and its components are technically able to deliver their value;
4. The *SecDevOps* perspective, which is essentially an integration after the Producing perspective and the Operating perspective with a focus on security, for organizations that develop and use home-grown software or home-grown extensions to standard software.

4.2 Perspective: Producing

Code reuse is an integral part of modern software. Specific software behaviour is still programmed of course, but for many years now it is standard development practice to rely on application frameworks and other 'plumbing' libraries for common functions and integrations. There is an abundance of application frameworks and reusable components out there, for any programming language and any technical runtime environment.
The widespread use of CI/CD (Continuous Integration/Continuous Development) pipelines greatly accellerates software development by automating much of the building of internal 'component repositories' in which frameworks and libraries reside. These repositories exist at least partly beyond the walls of the software producing organisation and all kinds of varying dependent component libraries are automatically, and often unconsciously, imported into the local repository when needed by a software function being programmed. It is in the 'unconscious' part where the SBoM adds significant value.

Figure 6 Usage of the SBoM in software development

4.2.1 Suggested Inclusion of the SBoM in Software Development

To ensure consistent transparency, an SBoM should be made available for any released version of a software product, as part of product documentation. Within the standard practice of CI/CD in the 'manufactoring' context of software production, the information to include in the SBoM is continuously updated. We advise to include the updates to SBoM information into the Definition of Done (or the organizations development approach equivalent of the DoD) and to include the availability of an SBoM as a test case in testing efforts. We also advise to include SBoM information automated updates to be a standard part of each build cycle in the CI/CD pipeline.

4.2.2 Considerations on Using the SBoM for Security Enhancement at development time

One of the more unfortunate results of automatically built library repositories, is that many different versions of the same library get imported into your component repository. This happens because based individual components each have their own dependencies on different underlying library versions. However, all those versions will have their own security vulnerabilites so one of the ways to improve code security is to reduce the variety of related component versions. The software transparency that an SBoM delivers provides knowledge of the level of technical variation underlying your software's implementation. Such knowledge allows you to limit that variation by reducing the variety of component (e.g. code library) versions and even block usage of specific components (or versions) which are known to have security vulnerabilities. Such knowledge also allows proactive assessment of vulnerabilities in your code, inherited through component dependencies.

Generating an SBoM automatically requires some careful considerations. The build system needs to be adapted to explicitly include SBoM information and even generate SBoM documents. A decision must also be made what to put into an SBoM (level of detail) and in what format. Chapter 5 in this report dives deeper into the information content aspects of an SBoM.

It is important to note that automated generation may lead to BIG SBoMs, since there is no real physical or time limit on the effort to generate it. Big SBoMs contain much information. But development teams still consist of humans, who will need to make informed interpretations of the SBoM contents. There will be a tradeoff between SBoM size and time alotted to put it to effective use.

4.3 Perspective: Choosing

The process of choosing a software component usually employs many steps of information gathering and selection. An important consideration should always be how the software component may impact the security posture of the organisation by having security vulnerabilities. An SBoM can contribute to the security posture impact assessment by allowing preliminary validation of such vulnerabilities.
The steps in the choosing process are not continuous but have a 'one-off' characteristic. Preliminary validation is achieved by ad-hoc matching of a software components' SBoM against vulnerability listings in known CVE repositories.
Ad-hoc matching can be done manually but it is not difficult to envision that manual matching quickly becomes combersome, since both SBoMs and CVE repositories can be large. Therefore we foresee that effective ad-hoc vulnerability matching is supported by digital tooling which provides visual ways of flagging potential vulnerabilities. This can even be done down to the source level (if the SBoM contains sufficient information to pinpoint the sources) to verify the trust level of source code providers underlying the software components embedded integrations.
The identified vulnerabilities can then be taken into account to form a balanced view of the security posture impact. Decisions can be made if additional (infrastructure) security measures should be taken, if specific installation planning is needed and if additional security integration testing should be employed before promoting the chosen software component to an operational production environment.

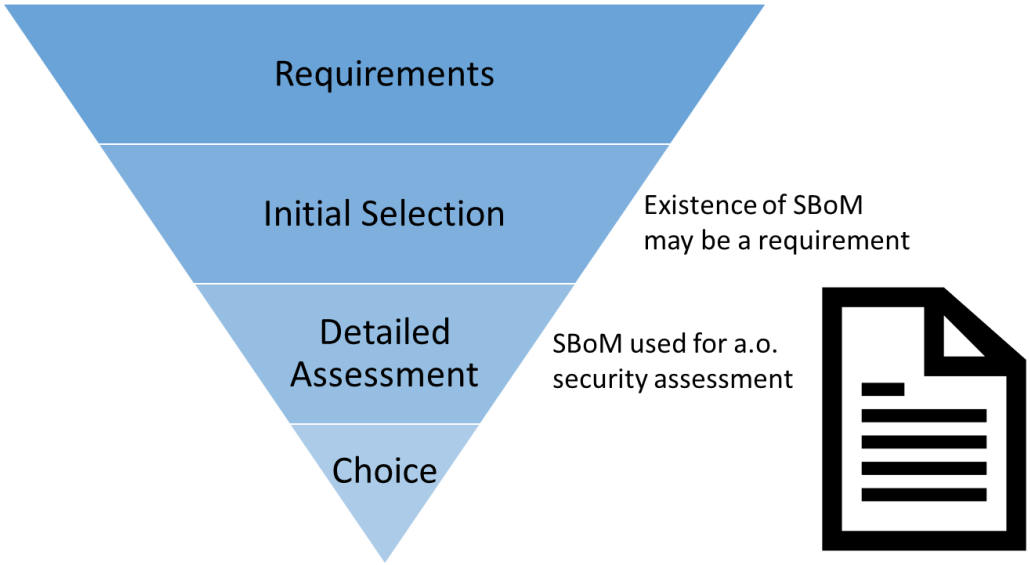


Figure 7 Usage of the SBoM when choosing IT components

Diving a bit deeper in the inner workings of digital tooling, we have to account for the lack of intuition of the average computer system. A computer must be able to unambiguously conclude 'yes this is a match'. Digital ad-hoc matching will thus require the use of standardized and correlatable data elements of SBoMs and CVE repositories, contained in agreed data formats and with validatable completeness, and containing compatible data values.

4.4 Perspective: Operating

The amount of security posture impact by software component vulnerabilities is related to usage spread of the software component in the organization's IT landscape. An important aspect of Secure Operations is the ability to continuously assess the IT landscape for potientially vulnerable software components. This assessment ability has the added benefit of making the organization's security posture independent of vulnerability information supplied by the software component supplier. The organization can mitigate any risks on their own accord.

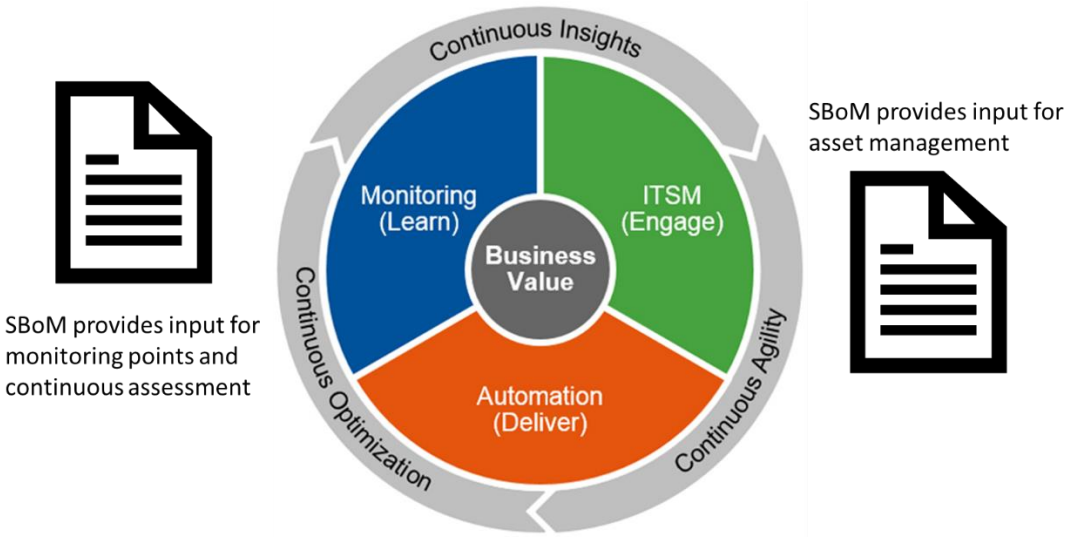


Figure 8 Usage of the SBoM in Operations

Continuous assessment implies a high level of automation. The information in an SBoM enables such continuous assessment by providing part of the data as input for that automation. Continuous assessment requires automated matching of SBoM information of software components against CVE repositories. Automated matching requires automation tooling, which matches standardized and correlatable data elements of software components SBoMs and CVE repositories,
contained in agreed data formats and with validatable completeness, and containing compatible data values.

Note that the SBoM data requirements of the Operating perspective are similar to those of the Choosing perspective. Both perspectives need to make computer-assisted assessments about software component vulnerabilities, only the level of automation differs.

4.5 Perspective: SecDevOps

SecDevOps essentially combines the Perspectives on Producing and Operation, mostly regarding in-house software development. DEVops, where it's all about developing the software, is linked to the Producing Perspective. Where the devOPS is linked to the Operation, because it is about bringing the (new) software into the operation through release, deploy, operate and monitor.
SecDevOps is not directly linked to the Choosing perspective, because you build your own software and solutions. You choose solutions but not software.

4.5.1 Suggestions on Using the SBoM during development (DEVops)

An SBoM would be made available for any released version of a software product, as part of product documentation. Within the standard practice of CI/CD, the information to include in the SBoM is continuously updated.

It starts with integrating security within the requirements. It can also be that some elements of SBOM are already added to a story or use case during the requirement phase. In this way, security and SBOM can become a standard and structured part of the DevOps process.
During refinement, and afterwards, discussions on solutioning will take place which will be based on best practices from the field (OPS). This can be a standard part of SBOM; which solution/technology/library/components are used. This can be edited per sprint, when necessary.
There should be a standard approach on implementing security solutions and/or technologies (within the functionality that is build). This could be, for example, the use of whitelisting instead of blacklisting, the use of specific libraries (or the libraries that you can't use!), charactersets, etc. This information is all added to the SBOM information and, after it's updated, will go with the team to the next sprint.
Also, it is important to use standard secure coding guidelines. SBOM can help to make code easier to review, because these coding guidelines may be based on existing frameworks that are connected to, for example a tool or specific libraries.

Testing of software components, either manually or automated, should be done through a structured approach. It is important that certain parts of the SBOM information will be tested every sprint as well. Are there libraries, components or technologies used that are vulnerable? There should be standard testing for abuse cases next to the usual functional testing of use cases.
In case of a pentest, the test of one application can also lead to the fix of other applications. You will know, because of the SBOM, that vulnerabilities exist here as well. You get a better overview of which components are used in which applications and if one has a vulnerability, the other ones who use these components will be vulnerable too. The advantage here is that you only have tested one application. Comparing SBOMS per applications must be a standard way of working.

To ensure continuous consistency, it makes sense to include the updates to SBoM information into the Definition of Done (or the organizations development approach equivalent of the DoD) and to include the availability of an SBoM as a test case in testing efforts. It would also be good practice to include SBoM information automated updates to be a standard part of each build cycle in the CI/CD pipeline.

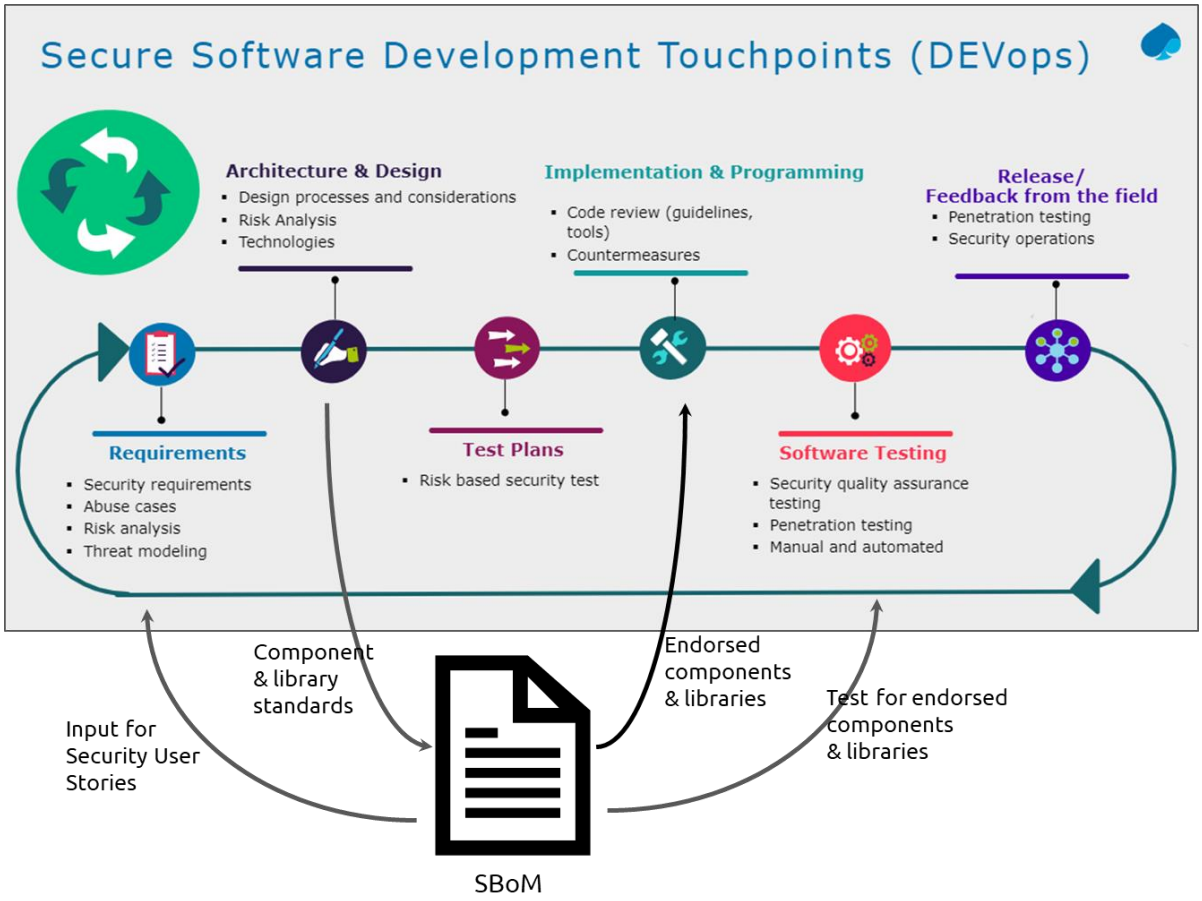


Figure 9 Usage points of the SBoM in DEVops

4.5.2 Suggestions on Monitoring and editing the SBoM during operations (devOPS)

During operations, the SBOM should be monitored and updated on the findings discovered. These could be based on feedback from the field, but they can also be discovered through monitoring.

A benefit of providing an SBOM to a customer is that you have a standard checklist and an overview of the used components in your application or application landscape. In case of an issue, fixing can go faster because of the documentation on the different components. And, issues are discovered earlier. If, as stated in the above paragraph as well, a library or component used has a vulnerability, you can easily check which application uses this and think of a general fix for all applications (if possible). This can lead to reducing costs through more streamlined and efficient administration.

In the paragraph about development, we already mentioned testing. During operations mostly pentesting is done on a structured basis. But also before release (or deployment) there should be a certain understanding of the (new) software's integration. You should have a clear overview of all linked components and the integrations they have or the integrations that exist between applications. Within the Secure Software Development Lifecycle (S-SDLC) in most cases there will be an integration test. SBOM, as also mentioned in the previous paragraph, can help connect vulnerable components in an easy and fast way. During operations monitoring these components on vulnerabilities is added to that.

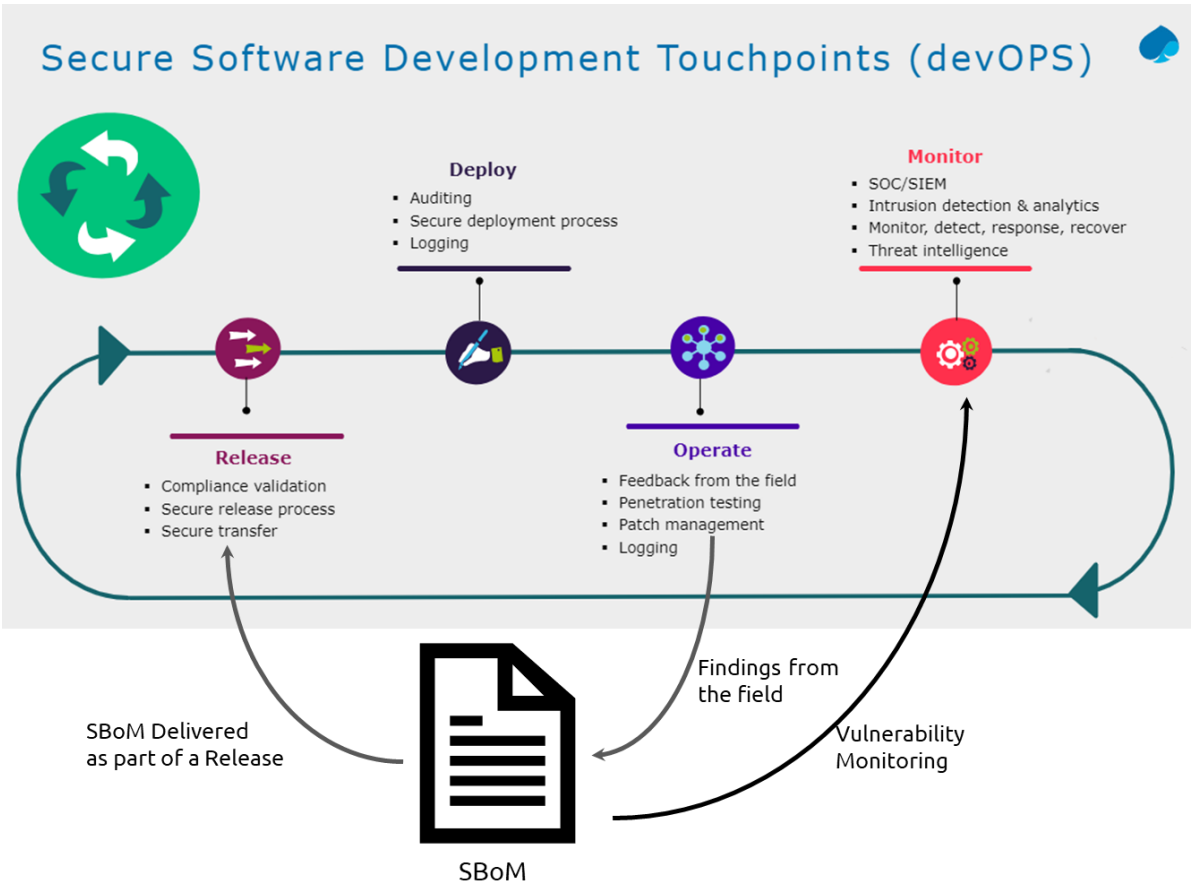


Figure 10 Usage points of the SBOM in devOPS

5. Key Success Factors for Vulnerability Detection with an SBoM

5.1 Online and offline availability of the SBoM

The optimal way of making an SBoM available depends on the usage scenario, but in general the SBOM should contain up-to-date information so any mechanism that supports keeping the SBOM up-to-date continuously has preference. An SBOM can be made available electronically for example via download to be obtained whenever the information is needed. Thinking in a more integrated fashion, the SBOM data could be made available via an exposed API to be integrated into for example build systems or asset management systems. For embedded software in devices which are deployed in a distributed manner, it may make more sense to package the SBoM with the embedded software on the device so that it may be consulted at will in circumstances where online integrations are not available – naturally it would not be kept up to date continuously in those circumstances.

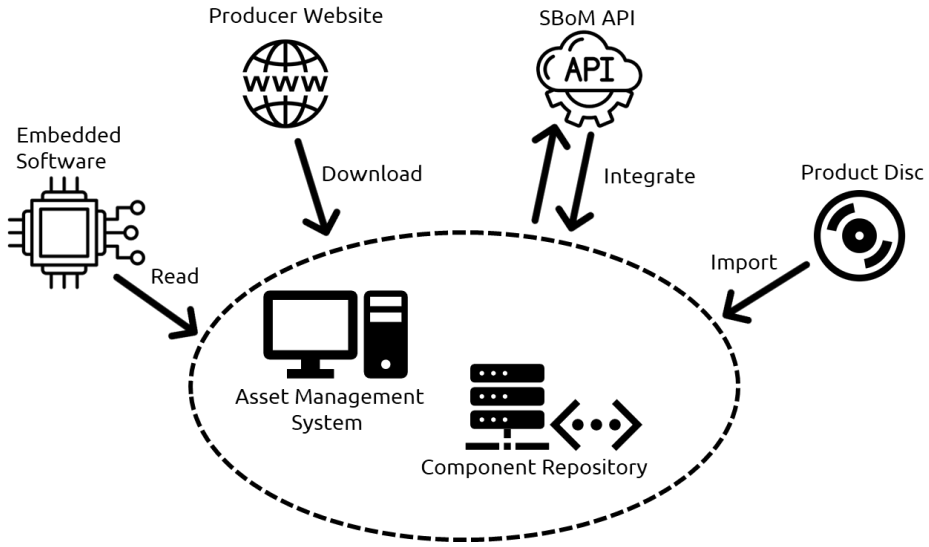


Figure 11 An overview of the methods to make an SBoM available

5.2 Fit for Purpose SBoM Information

The information contained in an SBOM is used differently in different scenarios. Chapter 2 describes that the usage of the SBOM to enhance security, differs between the four perspectives of Producing, Choosing, Operating, and SecDevOps. Therefore one of the key success factors for enhancing security is to have fit for purpose information in the SBOM. We can distinguish four aspects to having fit for purpose information:

1. Level of Granularity
2. Information Content
3. Data Standards
4. Data Formats

Level of Granularity: The Level of granularity is driven by the intended 'depth of control' of your software chain. Deeper control requires more granular information. Example: if you want deep insight into all possible vulnerabilities, you'll likely want to have each dependent component with its minor version and patch level listed. Whereas if your ambition is to facilitate portfolio management, you'd only need the top-level software component, its version and its supplier.

Information Content: Information Attributes must enable the security position you want to achieve. Different security controls require different types of information. E.g. simplified codebase by reduction of code bloat requires component identification and version information. E.g. a vulnerability impact assessment requires information of component usage relationships within IT landscape ('which applications are hit?').

Data standards: Automated correlation (e.g. with a CvE database, or a CMDB) requires standardization on data values and formats. Regarding data values, there is at least a need for an authorative source for Component Name, Version and Supplier.
In case of a device with embedded software, identification of the device should be included with the SBoM.

Data Formats: An SBoM data format should be self-describing, and structured, using a schema syntax and a name-space mechanism. The structure should also be self-validating by clearly marking required data attributes.

5.3 SBoM Information attributes required for security assessment

An SBOM document should provide the information for relating each component back to the broader instantiation. If there are relationships of links amongst the components, this is another beneficial data point to hold in an SBOM. Machine-readability is another key part of a useful SBOM document to allow for digitised and even fully automated processing of SBOM information to support software governance and IT security activities.
The primary purpose of an SBOM record is to uniquely and unambiguously identify components and their relationships to each other. In order to do so, some combination of the following baseline information is required[7].

- Author Name
- Supplier Name
- Component Name
- Version String
- Component Hash
- Unique Identifier
- Relationship

Weather you assess software component from their abilities manually or automated both approaches require an unambiguous correlation between the software component and vulnerability information. This implies that both the SBOM and the vulnerability entries must contain a unique identifier for that particular software component type and version. Additionally, information must be included in an SBOM to indicate relationships between this software component and any underlying other software component this component depends on.

CVE (Common Vulnerabilty Enumerator) is a data Holder format used to contain basic information about their vulnerability and provide links to additional information about that vulnerability from for example the software creator or additional sources on the Internet. CVE's often include links to CWE entries (Common Weakness Enumerator). CWE is a community maintained taxonomy of IT weakness types which is used to provide additional categorizations to vulnerabilities. There are several sources of CVS on the Internet mostly hosted by government or government funded institutions and providing so-called CVE dictionaries. A CVE dictionary is a searchable list of known vulnerabilities. A well known example in the United States is the cfe list of the NVD or national or vulnerability database. An often used CVE dictionary in Europe is CVE-Search by Luxemburg based entity CIRCLE.LU. CVE dictionaries

[7] https://www.ntia.gov/files/ntia/publications/ntia_framing_sw_transparency_v4.pdf, "Mapping to Existing Formats"

can be searched via web forms for human use, and often provide APIs to be searched by automated systems.

5.4 CycloneDX as preferred SBoM Data Format

Although there are many data standards to record software traceability and which could be used in an SBoM. Unfortunately, most of them do not natively include data attributes to hold vulnerability information. Such a data standard would naturally need to include both a unique identifier of the software component, and well-formatted vulnerability information.

SWID is intended to unambiguously identify software components with the goal of supporting managed deployment and IT landscape governance. The SWID format does include attributes to hold vulnerability references.

CycloneDX (https://cyclonedx.org/) can combine the unambiguous software component identity (using SWID attributes) and their relationships to other software components (using SPDX-like attributes). CycloneDX contains a vulnerability schema extension to hold URL references to CVE entries. CycloneDX essentially combines attributes of SWID and SPDX with the addition of explicit vulnerablity information. Identifying known vulnerabilities in components can be achieved through the use of three fields: `cpe`, `swid`, and `purl`. The table below lists the relationships between component types and the field to use. The CPE specification was designed for operating systems, applications, and hardware devices. CPE is maintained by the NVD and has been deprecated. Software ID (SWID) as defined in ISO/IEC 19770-2:2015 is used primarily to identify installed software and is the preferred format of the NVD. Package URL (PURL) standardizes how software package metadata is represented so that packages can universally be located regardless of what vendor, project, or ecosystem the packages belongs to.

Type of component to identify	Recommended Field Type
Client or Server Application	CPE or SWID
Container	PURL or SWID
Firmware	CPE or SWID
Library or Framework (package)	PURL
Library or Framework (non-package)	SWID
Operating System	CPE or SWID
Operating System Package	PURL or SWID

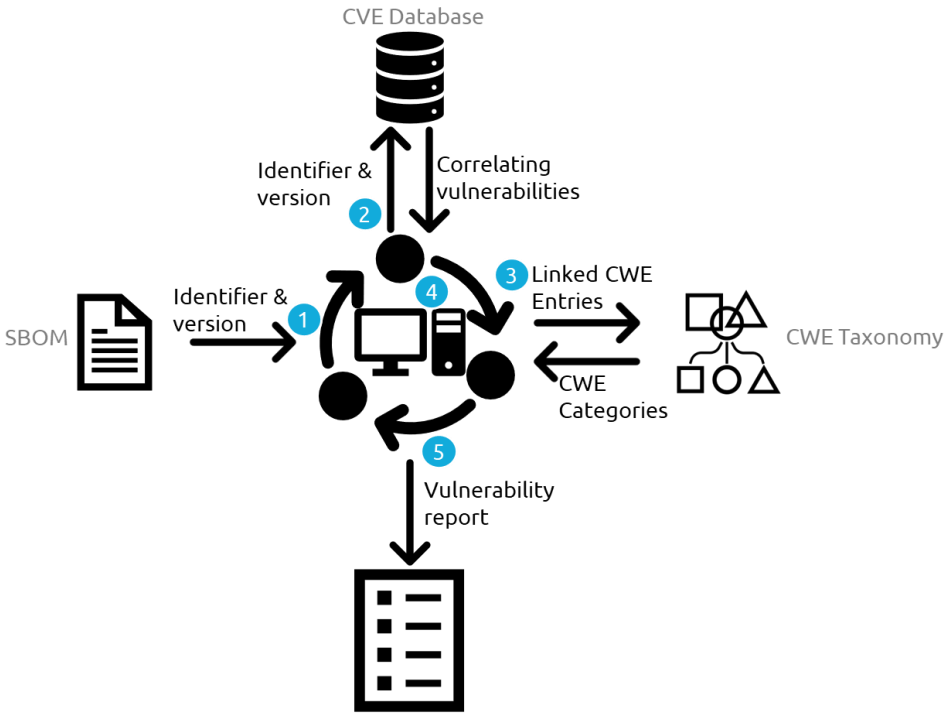
From a digitisation and automation point of view, CycloneDX would be the format of choice for producing SBoMs which are machine-readable. The CycloneDX specification defines both XML and JSON as supported layout formats. The content specification defines the required set of data attributes for vulnerability identification use cases. It is also a community maintained open standard which is actively developed, ensuring long-term support and usability.

5.5 Automation and Tooling to Enhance the Security Position

To exemplify the association of vulnerabilities to components, it is necessary to be able to correlate the software component identification (e.g. name and version) with one or more CVE's. The following two generic SBoM-Vulnerability automation scenarios can be distinguished.

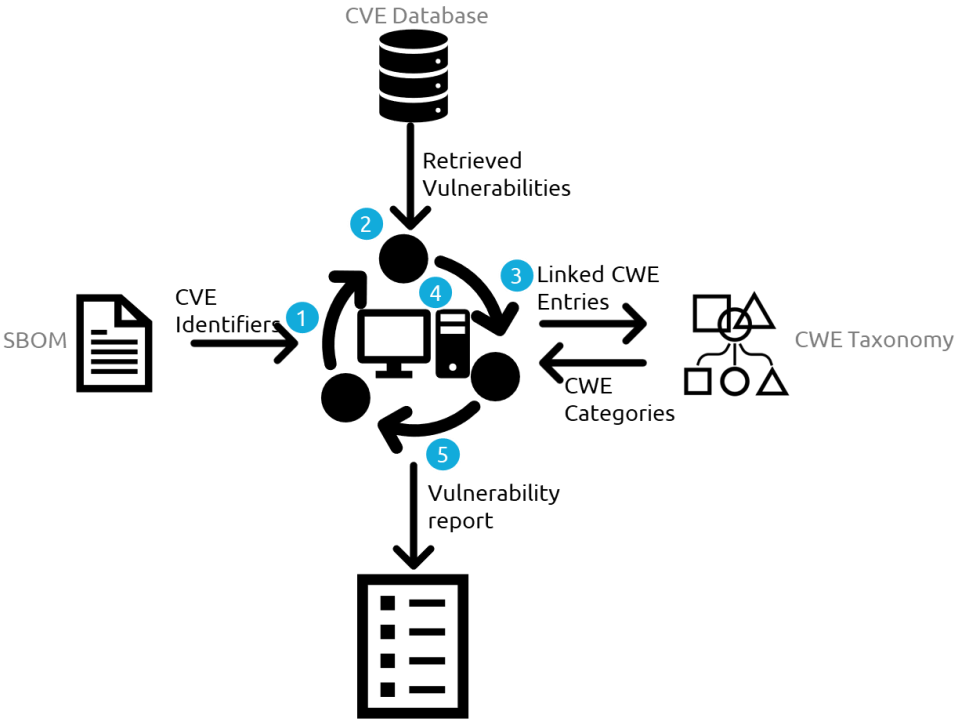
Scenario 1: Use SBOM component identifier for CVE dictionary search retrieve the CVE + associated CWE listings

1. Retrieve software component identifier + version from SWID/SPDX data
2. Search CVE dictionary for correlating vulnerabilities regarding the specific software component identifier and version and retrieve each found CVE entry
3. Retrieve CWE entries linked with the CVEs to be used as category flags
4. Repeat for each CVE entrie available for the specific software component ID and version
5. Generate the vulnerability report



Scenario 2: Use Vulnerability CVE links in the SBOM to directly retrieve the CVE + associated CWE listings

1. Retrieve CVE identifier from CycloneDX data
2. Retrieve the CVE entry from the CVE dictionary
3. Retrieve linked CWE entries to be used as category flags
4. Repeat for each CVE identifier in the CycloneDX data
5. Generate the vulnerability report



5.5.1 Tooling Requirements

At the very least, tooling should be able to:

- Import an SBoM into its internal database
- Perform ad hoc searches and correlation matching against CVE and CWE databases
- Generate a vulnerability report in which the information from the royalties is presented in a human readable way period

To be able to support continuous vulnerability detection w.r.t. software components tooling should also be able to:

- Periodically retrieve an updated SBOM from the software component producer
- Perform automated matching against CVS and CWS and continuously update digitized versions of vulnerability reports

in the context of Secure Software Development and SecDevOps, further integration with the CI/ CD pipeline would be very useful. For example, automation tooling could automatically download versions of software dependencies in which a vulnerability has been resolved, add those updated versions do the software component repository and directly integrate them into automated builds and testing to validate their compatibility.

5.5.2 Available Tooling

At the time of writing, the open source application DependencyTrack (from OWASP, https://dependencytrack.org) provides a reference implementation of the above functional requirements and uses CycloneDX as its base data format for SBoM.

6. Summary and Future Developments

6.1 Summary

The Software Bill of Materials can contribute to enhancing the security posture by facilitating continuous assessment of IT landscape vulnerabilities. Continuous assessment requires matching of SBoM information of IT components against vulnerability (CVE) repositories. The potentially vast amount of information within an SBoM and in vulnerability repositories means that continuous assessment can only be done using automated matching, using automation tooling. The tooling matches standardized and correlatable data elements of IT component SBoMs and CVE repositories.

Standardized data requires agreed data formats and validatable completeness of SBoM data. correlatable data requires compatible data values between SBoM data and CVE repositories. Such a data standard would naturally need to include both a unique identifier of the software component, and well-formatted vulnerability information.

6.2 Future Developments

The Software Bill of Materials is clearly gaining traction, but accepted data standards, tool support and applicable best practices seem to be limited. A natural next development would be towards further standardization and enhanced tools support. Also, the options for applying certification schemes to the software bill of materials as a concept should be considered.
The potentially overwhelming amount of data to enable of vulnerability correlation using an SBoM makes this a likely candidate for effective use of artificial intelligence (AI). As a next step, a pilot could be set up for actual usage of the SBoM in vulnerability analysis using AI. As part of the pilot, new ideas on tooling functionality and data matching algorithms could be explored as well as well as gaining new insights on potential improvements for business processes.
The future path of the Software Bill of Materials can only be expected to materialize whith a combination of active involvement public institutions and participation by private organisations. Public bodies should provide the regulatory and standardisation context which gives clear direction to the development of the SBoM. Private organisations should then take the initiative to make the initial investments into the realisation of standardized software transparency tools, data standards and business processes to reap the security benefits of the SBoM.

7. Appendix

7.1 Specialists Capgemini

Bart van Riel is a Capgemini consultant with the role of Managing Enterprise Architect. His focus is on harmonizing and balancing the human and automated factors within organizations. Bart has extensive experience within the international Cyber Security domain. He is also a mentor for aspiring architects within Capgemini and client organizations.

Sanne Kuijpers is a Capgemini SecDevOps specialist. She leads the SecDevOps core team within Capgemini NL. Sanne is in direct contact with our customers and is often working as a trusted advisor on site. She aims to make security a structural part of the SDLC. Her focus in this is on the team, because the mindset and motivation of the team are key to all the processes.

Roeland de Koning is a Capgemini consultant and specializes in cybersecurity and crisis management. He focuses on bringing about both national and international collaboration when it comes to these issues.

7.2 Respondents

- Nicole van Deursen NCSC
- Brenda Langedijk Agentschap Telecom
- Leo Groenendaal Erasmus MC
- Ben Kokx Philips Medical
- Bart van Riel Capgemini
- Sanne Kuijpers Capgemini
- Barry Jones Capgemini
- Bob Ranzijn Capgemini
- Maikel Ninaber Capgemini
- Vincenzo Corona Capgemini

7.3 Bibliography

Description	URL
Existing standards & formats SBoM	https://www.ntia.gov/files/ntia/publications/ntia_framing_sw_transparency_v4.pdf
Cyber security standards in the US	https://www.congress.gov/bill/113th-congress/house-bill/5793
Uses cases SBoM	https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf
Proof of concept of SBoM for healthcare	https://www.ntia.gov/files/ntia/publications/ntia_sbom_healthcare_poc_report_2019_1001.pdf
Presentation from PoC SBoM Healthcare	https://www.ntia.doc.gov/files/ntia/publications/ntia_sbom_healthcare_poc_2020-07-09.pdf
Overview from the NTIA	https://www.ntia.gov/SBOM
Cyclone DX example	https://github.com/CycloneDX/specification
SPDX contents	https://spdx.dev/wp-content/uploads/sites/41/2017/12/spdx_onepager.pdf
Alternative format (SID Tags)	https://www.iso.org/standard/65666.html
SWID tags	https://www.snowsoftware.com/blog/2014/03/13/swid-tags-what-are-they-and-why-are-they-important
Overview literature	https://www.ntia.doc.gov/files/ntia/publications/ntia_sbom_formats_and_standards_whitepaper_2019_0904.pdf
Overview literature	https://www.ntia.doc.gov/files/ntia/publications/ntia_sbom_roles_and_benefits_06.25.pdf
Overview literature	https://www.synopsys.com/blogs/software-security/software-bill-of-materials-bom/
How to implement SBoM yourself	https://promenadesoftware.com/blog/create-a-software-bill-of-materials
Why SBoM does not work for SecDevOps	https://cybersecurity.att.com/blogs/security-essentials/software-bill-of-materials-sbom-does-it-work-for-devsecops
Best practices	https://owasp.org/www-community/Component_Analysis