Consolidated SBOM and CSAF/VEX Operational Framework

Version 1.0

Jun 23, 2023

Consolidated SBOM and CSAF/VEX Operational Framework

1 Executive summary

The supply chain of the technology ecosystem presents an excellent opportunity for adversaries and malicious actors. Supplier codebases, whether commercial third-party or open source software components, are a potential conduit for the distribution of malicious payloads and security vulnerabilities downstream to their consuming products. To combat this issue, vendors in the technology ecosystem are striving to embrace and implement industry practices like the Software Bill of Materials (SBOM), Common Security Advisory Framework (CSAF), and Vulnerability Exploitability eXchange (VEX) to inform the consuming organization of content, scans, known and resolved issues, and best practices. These practices are essential for improving software security, enabling effective vulnerability management, and fostering transparency and trust between vendors, open source communities, and customers.

SBOM is a structured inventory or manifest of software components and dependencies used in a particular software product and is a static file associated with a particular software release. It allows producers and vendors to identify and track the origin of third-party and proprietary components. It provides the ability for ease in procurement and audit operations while also enabling efficient vulnerability management. By implementing SBOM, vendors can enhance the security posture of their products and mitigate potential risks associated with vulnerable or outdated software components. Customers benefit from SBOM by gaining visibility into the software supply chain, where it originated, and the licensing that flows from the supply chain. This information enables them to make informed decisions regarding the use of software products.

CSAF is a standardized format for sharing security advisories and vulnerability information. By adopting CSAF, suppliers can provide timely and accurate information about security vulnerabilities and known exploitation in their software releases. This allows customers to assess and prioritize their risk and vulnerability management efforts, ultimately improving their ability to protect their systems and data. Vendors benefit from CSAF by establishing a consistent and efficient industry means of communicating security advisories, which will reduce response times and build trust with their customers.

VEX is an open, neutral platform that facilitates the exchange of vulnerability information between vendors and customers. It is the set of data released dynamically upon discovering issues that are fixed and then published. By leveraging VEX via CSAF, vendors can efficiently share vulnerability data, patches, and remediation guidance with customers, streamlining the vulnerability management process. VEX provides customers with a centralized repository of vulnerability information, keeping them informed and updated on the latest vulnerabilities affecting their software ecosystem. By participating in VEX, vendors and customers can collaborate in real time, fostering a community-driven approach to vulnerability management. VEX is an open, neutral platform that facilitates the exchange of vulnerability information between vendors and customers. It is the set of data released dynamically upon discovering issues that are fixed and then published

Implementing SBOM and VEX via CSAF (CSAF/VEX) brings several important benefits to both vendors and customers: the suppliers and consumers in CSIA/NTIA language. By adopting SBOM, vendors can efficiently manage the software supply chain and ensure the integrity of their products. CSAF enables vendors to communicate vulnerabilities effectively, facilitating quick and informed decision-making by customers. VEX provides a collaborative platform for vendors and customers to exchange vulnerability information and remediation guidance, improving security outcomes. VEX is dependent on a vendor implementing CSAF to provide full details to a consuming organization.

Customers benefit from the implementation of SBOM and CSAF/VEX by gaining transparency and insight into the security posture of their software products. SBOMs allows customers to make informed decisions based on the software supply chain information. CSAF provides standardized and timely security data, enabling customers to prioritize their risk and vulnerability management efforts and protect their systems effectively. VEX facilitates the real-time delivery of security data to customers facilitating collaboration with vendors for prompt attention.

By embracing SBOM and CSAF/VEX, vendors can enhance knowledge of software released, security information, and foster a more secure software ecosystem. Customers, on the other hand, can make informed decisions, prioritize their vulnerability management efforts, and stay ahead of emerging threats. Together, these standards and frameworks pave the way for a more secure and resilient software ecosystem.

This living document guides vendors on implementing and releasing SBOM and CSAF/VEX information. Updates to this document will occur as these practices evolve and the roadmaps and strategies of technology ventures are implemented.

2 Introduction

2.1 Purpose

The Operational Framework for SBOM and CSAF/VEX is a high-level document which provides recommendations and guidance to organizations creating (suppliers) systems that support the distribution of Software Bills of Materials (SBOM) for their products. This also enables the automation of risk management for their customers through the development of machine-readable security advisories which follow CSAF incorporating VEX guidelines.

These recommendations were developed by recognized PSIRT organizations and vulnerability management experts from industry-leading organizations to provide a more consistent experience for our customers and the ecosystem at large.

This Operational Framework focuses on the following customer objectives and goals for vendors implementing these practices:

- Gain a better understanding of the source of a product's underlying components in order to identify components that have security vulnerabilities more quickly and efficiently to mitigate those vulnerabilities. This is of particular importance when security addendums have been added to supplier contracts; Reference: PSIRT Framework 1.3.1.
- Gain better insights into the vulnerabilities exploitability that exist within deployed products, enabling better prioritization, faster response times, and reduced costs for vulnerability management.
- Increase accuracy regarding the impact or severity of a vulnerability to a given vendor's product or component. For example, some vulnerabilities are only exploitable when a specific condition is met within a component, these are often incorrectly identified by vulnerability scanners as true positives. This is notable for Open Source Software where backporting or rebasing of code often takes place.
- Increasing the speed, accuracy and status of known exploitation against identified components and applications.
- Improve security posture through the ability to proactively inform customers, working with vendors to address known vulnerabilities and establish a predictable cadence for updating.

2.2 Definitions

Advisory: Announcement or bulletin that serves to inform, advise, and warn about a vulnerability and associated mitigations and fixes that affect a product.

Disclosure: The act of initially providing vulnerability information to a party that was not believed to be previously aware. The overall disclosure process typically includes multiple disclosure events and coordinations.

Exposure: The time between the discovery of a vulnerability and the time a fix for a vulnerability is released so that it may no longer be exploited.

Mitigations: Actions that reduce the likelihood of a vulnerability being exploited or the impact of exploitation.

Remediation: Patch, fix, upgrade, configuration, or documentation change to either remove or mitigate a vulnerability.

Vendor: Individual or organization acting as the supplier that developed the product or service or is responsible for maintaining it.

Vulnerability: Weakness in software, hardware, or a service that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or group of components. A vulnerability normally has a Common Vulnerability and Exposures (CVE) number assigned to identify it.

Software Bill of Materials (SBOM)
SBOMs have existed for many years to ensure that supplier codebase is released under a license in which the copyright holder grants users various rights, including the rights to use, modify, and distribute the software and its source code or binary release, which is made available publicly. Supplier codebase is copyright protected, and its use requires compliance with the associated license. While focused on addressing licensing violations, SBOMs have become the foundation to provide vendors greater transparency into the components embedded in their supplier's codebase.

The National Telecommunications and Information Administration (NTIA) of the U.S. Department of Commerce released the following definition of an SBOM:

> *An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships.*

> *These inventories should be comprehensive – or should explicitly state where they could not be. SBOMs may include open source or proprietary software and can be widely available or access-restricted.*

An SBOM is a nested inventory that includes all the components that make up a given software product and all of the components that make up those components, to the degree to which the vendor can confidently assert.

SBOMs can serve different purposes depending on the context in which it is used. An SBOM that shows compliance to open source licenses that may be susceptible to legal risks focuses on identifying all the open source components used in a software product and their associated licenses. It helps ensure that the software product is in compliance with the licensing requirements of the open source components and avoids legal risks associated with non-compliance. This type of SBOM is typically used in the context of software supply chain management and is often required by customers or regulators as part of the procurement process.

On the other hand, an SBOM used for the intent of vulnerability management focuses on identifying all the components and dependencies of a software product and assessing their potential security vulnerabilities. This type of SBOM helps organizations prioritize and manage the remediation of vulnerabilities in their software products. It is used in the context of security risk management and is typically part of an organization's vulnerability management program.

In summary, while both types of SBOMs list the components and dependencies of a software product, they differ in their intended purpose. An SBOM for compliance focuses on open source licensing requirements and legal risks, while an SBOM for vulnerability management focuses on security risks and the management of software vulnerabilities. Since SBOMs do not convey security data, it is a list of components in a release or manifest that provides provenance of the code, stating the organization or project that produced the code, where it was pulled from if from a third party or domain, and it includes the licensing for that code. Most SBOMs are used in procurement, audit and establishing the base for a risk and vulnerability management program. Without the CSAF/VEX data, it cannot provide a means to establish current risk or remediation.

In May 2021, United States President Biden issued Executive Order #14028 on "*Improving the Nation's Cybersecurity*," which included provisions requiring vendors contracting with the federal government to provide SBOMs with their products, adding urgency to the effort to define baseline components and acceptable formats for SBOMs.

Common Security Advisory Framework (CSAF)

ISO 29147 and PSIRT Services Framework call out the need for a security advisory to inform customers of security vulnerabilities within a vendor's software or hardware product. This communication method enables customers to make an informed decision. CSAF is a standard used to create machine-readable security advisories to allow the consumption of this information in an automated way. CSAF version 2.0 was developed as an open standard project of the Organization for the Advancement of Structured Information Standards (OASIS) and was officially approved as an OASIS standard in November 2022.

The framework defines primary properties, and many levels of nested properties, in order to support the standardization of security advisories across companies, products, and fields. By providing standardization for advisory creation and distribution, CSAF supports the automation of vulnerability management, which can decrease the time between vulnerability disclosure and a customer's implementation of remediations. CSAF provides comprehensive support for VEX and has been adopted by several key industry stakeholders and government institutions such as Cybersecurity and Infrastructure Security Agency (CISA) and The Federal Office for Information Security (BSI). CSAF supersedes the Common Vulnerability Reporting Framework (CVRF), which was originally created by Internet Consortium for Advancement of Security on the Internet (ICASI).

Vulnerability Exploitability Exchange (VEX)

A VEX document is a type of security advisory developed by the U.S. National Telecommunications and Information Administration (NTIA) "*to provide users (e.g., operators, developers, and services providers) additional information on whether a product is impacted by a specific vulnerability in an included component and, if affected, whether there are actions recommended to remediate.*"

In CSAF, VEX is one of five available profiles. Each profile defines a specific use case for a security advisory and the fields required to achieve its purpose. The CSAF specification says, "*The main purpose of the VEX format is to state that and why a certain product is, or is not, affected by a vulnerability.*"

The products referenced in a VEX security advisory must have one of four statuses:

1. Not affected: No remediation is required regarding this vulnerability.
2. Affected: Actions are recommended to remediate or address this vulnerability.
3. Fixed: These product versions contain a fix for the vulnerability.

4. Under Investigation: It is not yet known whether these product versions are affected by the vulnerability. An update will be provided in a later release.

The primary feature that distinguishes a VEX security advisory from other kinds of advisories is that a VEX advisory can be used to declare that a product is *not* affected by a vulnerability and why it is not affected. Although it is not essential for a product to have an SBOM to use VEX-type security advisories, VEX is designed to work with SBOMs. Some implementations have identified the SBOM signature to allow a direct correlation between them.

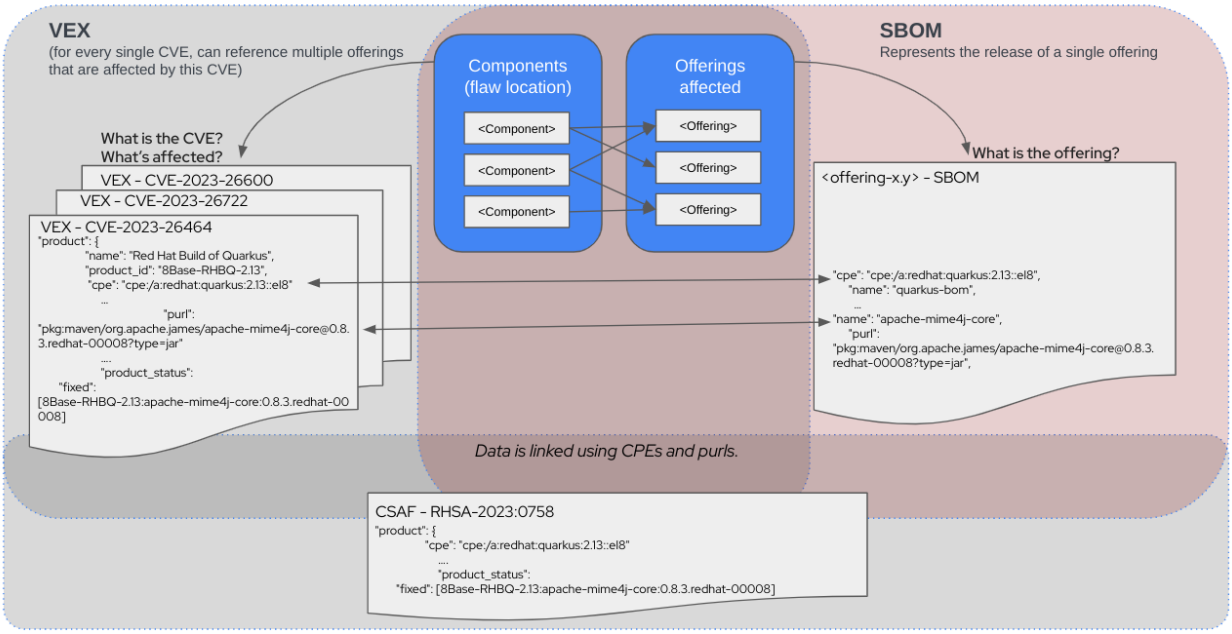
2.3 How SBOM, CSAF, and VEX work together

An SBOM provides a customer with an inventory of what components make up the software they use. With that information, an asset management system can track all disclosed vulnerabilities that *might* affect the software and its component parts. This setup does not provide an accurate, actionable risk assessment by itself. An SBOM allows a consuming organization to determine risk based on provenance or licensing requirements, but it does not resolve vulnerabilities or exploitations by itself.

CSAF and VEX help the customer by providing currently published vulnerability information for customers to act on. By distributing security advisories and exploitation data in the CSAF format, the vendor directly empowers the customers' asset management systems to find, download, and evaluate advisories and take action. The vendor uses the VEX profile within CSAF to inform their customers that a product is *not* affected by a vulnerability, allowing customers' asset management systems to disregard that vulnerability, thereby allowing administrators to focus on the vulnerabilities that could impact their businesses by using VEX statuses like *Affected* or *Under Investigation*. The *not affected* status can be due to several factors that the vendor or producer is aware of, from mitigations in the application, backporting or rebasing of the code, or that the version of the component is not affected by the vulnerability.

In isolation, each of these artifacts solve a particular problem. Together, SBOM and CSAF with VEX provide another level of transparency from or between suppliers and to the consumers. This approach provides the capability to create a direct vulnerability management ecosystem where the time between vulnerability disclosure and a customer's implementation of the necessary remediations, when the customer is most vulnerable to attack, can be drastically reduced. Even if a remediation to the supplier codebase is not available, the vendor can potentially provide other workarounds or mitigations through compensating controls to reduce the attack surface until the supplier codebase has been remediated. This is a direct exchange of security data for immediate use.

SBOM and CSAF-VEX relationship



2.4 Why does this matter to the industry and ecosystem?

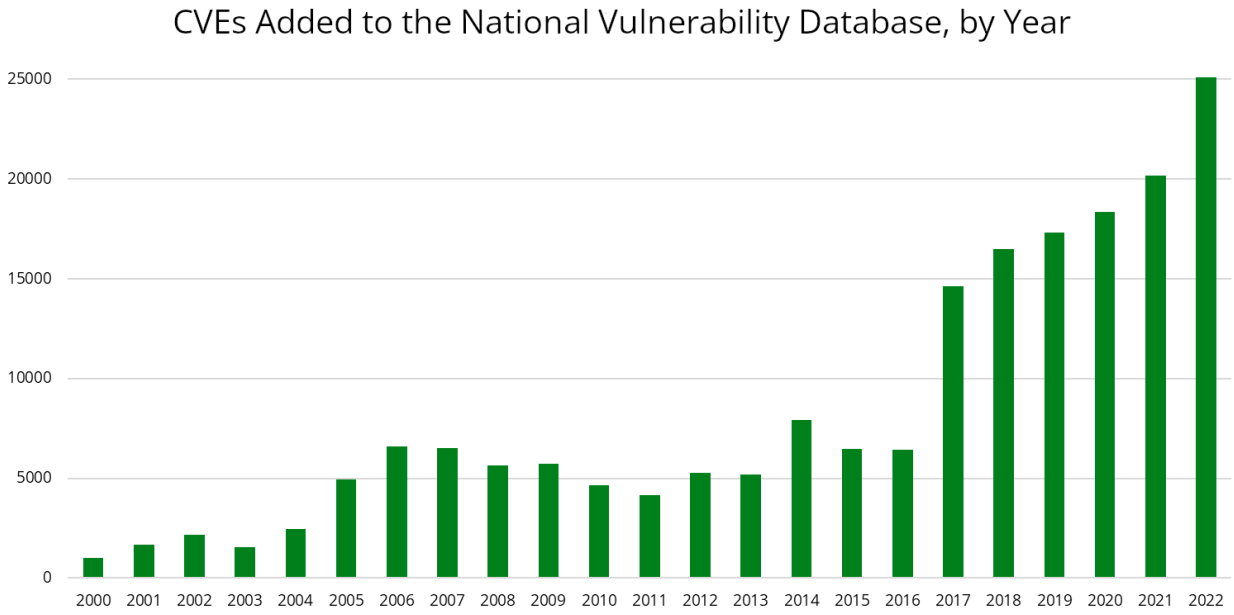
Third-party components are an enabler of technology, providing the capability for development teams to:

- Drive efficiencies
- Improve customer outcomes
- Achieve faster time to market
- Reduce development costs

The benefits of consuming commercial third-party components, proprietary and open source, have been reduced due to the high-profile public disclosures on components that carry significant risk, such as log4j, heartbleed, and Apache struts. We have found ourselves in the perfect "Supplier and Consumer Security" storm. This dramatic and ongoing increase in vulnerability issues are discovered and disclosed annually as aggregated by NVD shown in the

graph below

The closed solutions of yesterday no longer carry weight in today's environment. Transparency into our supplier's code combined with automation is the only path forward to effectively and quickly manage vulnerabilities at an industry-wide scale. The basis to enable this automation primarily comes from the proprietary and open source vendors and communities. The inability to manage vulnerabilities effectively because of the lack of automated solutions directly impacts our critical infrastructure and economies much more, along with those tasked with responding to such issues on behalf of their companies.



Using the community-based recommendations outlined in this document, organizations can create vulnerability management systems that automate the distribution of security advisories. Their customers, in turn, will be able to use automated asset management systems that collect, evaluate, and prioritize vulnerability information. This will empower customers to take action to remediate those vulnerabilities that pose the greatest risks to their businesses. The additional

benefits to organizations are that they also address procurement questions and satisfy some audit requirements on vulnerabilities, provenance, and licensing. That combination allows for a same approach with data, therefore, the previous approach of separate data that allows gaps dissipates.

3 Implementation guidance

3.1 Software Bills of Materials (SBOM)

An SBOM is a static file that includes the product, version, and release information, also known as the manifest. The manifest is a listing of the components used with the software offering and in some cases should also include the build dependencies of the release pipelines. An SBOM should include the provenance of the code as to where that code came from, for instance the project or entity that constructed the code and the provenance should also indicate where you pulled that code from as in the domain / url. Also included in an SBOM is the licensing information for use of that code. Together, these assist customers in answering their procurement questions and legal use for their customers downstream while also forming the mapping for a risk management program.

There are at least six different types of SBOM formats: Design, Source, Build, Analyzed, Deployed, and Runtime, and are commonly delivered using one of two formats: SPDX or CycloneDX. Once you create a system for compiling SBOMs for the products in your portfolio, you may want to consider having this available in both a machine and human-readable format. The machine-readable format will assist in evaluating and managing risk of security vulnerabilities within components used by your product portfolio and greatly aid in the distribution and consumption of the data. On the other hand, the human-readable format will be useful for procurement and legal teams to address questions related to provenance and licensing. The manifest element of the SBOM may also provide minimal support for risk assurance by demonstrating that the code has been scanned with anti-virus and anti-malware tools.

All of this is good information, but overall, it has confused the discussion for customer use in a general business conversation. Simply, the SBOM provides you with information regarding the code or product ingredients, nothing more. The 'SBOM' delivery or communication formats, SPDX and CycloneDX, have too many options, while also including both static and dynamic data sets. This has overly complicated the ability for general businesses to have quick adoption and long term sustainability as a whole. In other words, we have a technically elegant solution that confuses the usage, with SBOM becoming a catch all for SBOM, VEX, and Attestation, thus requiring a translator to gain succinct actionable knowledge. We propose that the SBOM is a

standalone static document from the producer, whereas the CSAF/VEX is a dynamic, changing document, that may be machine-readable from the producer, that allows a customer to pull the data they need in order to take quick action. Currently, attestations are primarily for government requirements and are not generally published to the public.

The lack of accurate and up-to-date SBOMs can expose organizations to security risks in their software supply chain management. Without a clear and common SBOM strategy, organizations may struggle to generate and maintain accurate SBOMs. As a result, there is a need for a set of recommendations to successfully implement an SBOM strategy that can ensure the accuracy and reliability of SBOMs for compliance and vulnerability management purposes.

The following are several recommendations by industry experts to successfully implement an SBOM strategy in your organization. By implementing these recommendations, organizations can ensure accurate and up-to-date SBOMs for compliance and vulnerability management purposes.

3.1.1 Basic implementation: what do you need?

As a consumer of an SBOM, your focus is primarily on a risk management program. Such a program needs to fully understand the hardware, software and firmware asset management within your operations / network. This involved hardware and software asset management, which will likely matrix to each other for quick incident response for prioritized remediation or compensating controls. From the software side, due to the ever growing amount of code in your network, the object now is to understand all code from all sources so that you can compare the components against vulnerabilities and known exploitation. Do not forget the code that you develop in house. For many organizations, this is about either trusting an external party to protect your assets or to build your own vulnerability management system to compare against all of your own known components that are deployed and where they are running in your environment. That is the essence of having both an SBOM (asset manifest) and to gaining

Vulnerability Exploitation eXchange (VEX) information to inform your asset management and response program.

It is essential to ensure consistency between licenses leveraged to address legal concerns versus those for security concerns. Misalignment may mean that there may be licensing inconsistencies. The Cybersecurity Resiliency Act supports this, which states, "*In order to facilitate vulnerability management, the NTIA has released a report that outlines the specific minimum analysis, where manufacturers should identify and document components contained in the products with digital elements, including the software bill of materials.*"

Vendors may produce SBOMs in either or both the SDPX or CycloneDX format; see Section 3.1.3 Recommended SBOM Formats. However, vendors are not required to generate SBOMs in multiple formats. A vendor can choose one format and later switch to a different format, which should not impact their customers.

It is worth noting that the CycloneDX and SPDX initiatives are always working towards compatibility with each other, allowing organizations to exchange SBOMs regardless of the format used. This is usually done with SBOM format translation tools. This interoperability enables greater flexibility in software supply chain management and supports the adoption of standardized SBOMs across industries.

For customers, free, open source, and commercial tools are available for asset management and SBOM consumption. Any asset management tool should be able to process SBOMs in both SPDX and CycloneDX formats. Customers should evaluate tools for all areas that support their risk management needs, such as license compliance, before choosing one.

Any asset management tool should be able to take the list of components from an SBOM and look up vulnerabilities that affect those components. A tool may ingest SBOM files using REST APIs, plugins, manual uploads through a web interface, or other methods. Customers should restrict access to SBOM and vulnerability information to those who require it for their work, such as cyber risk managers and incident responders.

Incident response is one area where SBOMs can be critical in helping customers understand their risk. For critical incidents such as Log4j, knowing what software in their environment is affected will allow customers to prepare to deploy out-of-cycle patches once they are available from their vendor. SBOMs will also help customers identify what products and components are potentially affected and what security data must be checked on the vendor side to complete the risk assessment because not every vulnerability affects all software vendors in the same way.

In addition to incident response, customers can use SBOMs to assist with software purchasing decisions and gain insight into the supply chain risk for software already deployed in their environments.

3.1.2 Creating an SBOM

As vendors turn their product manifests into SBOMs, they should take requirements and guidance from the NTIA publication, The Minimum Elements For a Software Bill of Materials (SBOM). According to this document:

> *The minimum constituent parts of an overall SBOM, referred to as elements, are three broad, inter-related areas. These elements will enable an evolving approach to software transparency, capturing both the technology and the functional operation... These three categories of elements are:*
>
> - *Data Fields*
> - *Automation Support*
> - *Practices and Processes*

3.1.2.1 SBOM Requirements Matrix

This matrix outlines the minimum elements for an SBOM and additional recommendations to consider beyond the minimum elements.

The minimum elements for a Software Bill of Materials

Req #	Element	
1	**Data Fields** Baseline information about each component that should be maintained in order to enable component tracking across the software supply chain and mapping to other beneficial sources of data, such as vulnerability databases or license databases.	
	Data Field	**Description**
	Supplier Name	The name of an entity that creates, defines, and identifies components.
	Component Name	Designation assigned to a unit of software by the original supplier.

	Version of the Component	Identifier used by the supplier to specify change in software from a previously identified version.
	Other Unique Identifiers	Other identifiers that are used to identify a component, or that serve as a look-up key for relevant databases.
	Dependency Relationship	Characterizing the relationship that upstream component X is included in software Y.
	Author of SBOM Data	The name of the entity that creates the SBOM data for this component.
	Timestamp	Record of the date and time of the SBOM data assembly.
2	**Automation Support** Support for automation should provide the ability to scale across the software ecosystem, particularly across organizational boundaries. This necessitates predictable implementation and data formats.	
		Description
	Automatic generation	Should Allow for scaling across the software ecosystem. SBOM files should be generated automatically at release and then become static files. VEX files should be generated automatically and released as vulnerability or exploitation status is updated
	Machine-readability	The data formats that are being used fo generate and consume SBOMs are: ● Software Package Data Exchange (SPDX) ● CycloneDX ● Software Identification (SWID) tags
3	**Practices and Processes** Vendors should integrate SBOMs into the operations of the Secure Software Development Lifecycle (SDLC). A number of elements should be explicitly addressed in any policy, contract, or arrangement to ask for or provide SBOMs.	
		Description
	Frequency	Vendors should issue a new or revised SBOM when: ● The software component is updated with a new build or release ● A software build integrates an updated component or dependency ● The vendor learns new details about the underlying components or wishes to correct an error in the existing SBOM data
	Depth	At a minimum, an SBOM must list all top-level dependencies with enough detail to seek out the transitive dependencies recursively. An SBOM consumer can specify depth by number of transitive steps or in operational terms.
	Known Unknowns	SBOMs must explicitly identify the components for which the presence of dependencies is unknown and incomplete. This must be integrated into the automated data.

	Distribution and Delivery	SBOMs should be available in a timely fashion to those who need them and must have appropriate access permissions and roles in place.
	Access Control	If access control is desired, the terms must be specified, including specific allowances and accommodations for integrating SBOM data into the user's security tools.
	Accommodation of Mistakes	Consumers of SBOMs should be explicitly tolerant of the occasional incidental error.

Beyond the minimum elements

There are also other recommendations to consider beyond the minimum elements.

Recommended Data Fields The following data fields are recommended for consideration, especially for efforts that are planned over several years or that require higher levels of security.	
Data Fields	**Description**
Hash of Component	A cryptographic hash provides a foundational element to assist in mapping the existence of a component to relevant sources of data, such as vulnerability data sources, and helps in instances of renaming and changes to allowed lists.
Lifecycle Phase	An SBOM should note how the data was captured, for example: source, build, or post-build, to help with consumption and data management.
Other Component Relationships	Other types of dependency relationships, derivation or descendancy, can be included in an SBOM.
License Information	SBOMs can convey data about the licenses for each component.
Recommendations for Cloud-based Software and Software-as-a-Service	
Cloud service providers should assert that they have an internal SBOM. That SBOM must be maintained with the rough functional equivalents of the minimum elements specified above.	
Cloud service providers must also have the capability to act on SBOM and vulnerability information and have a process to do so in a timely fashion.	
Recommendations for Ensuring SBOM Integrity and Authenticity	

Those supplying and requesting SBOMs should explore options to both sign SBOMs and verify tamper-detection. Such a mechanism should allow the signing of each component of a given piece of software and allow the user to determine whether the signature is legitimate.
Recommendations For Addressing Vulnerability and Exploitability
Vendors should track vulnerability data in data structures that are separate from the SBOM. Operations should focus on mapping and linking between the two types of data as each evolves and the technologies mature. If vulnerability data is shared across organizations, both the vulnerability data and the SBOM can use similar models for distribution, access control, and ingestion.
Vendors should communicate the status of a vulnerability to customers using VEX once they have made a reliable determination.
Customers should ensure that their tools for analyzing SBOM data are able to automatically incorporate VEX data.
Recommendations for Managing Legacy Software and Binary Analysis
Customers, especially those in charge of critical infrastructure, should use binary analysis tools on legacy software if SBOM data is not available. Binary analysis can also be used to validate SBOM contents or help understand gaps in the SBOM data.

3.1.3 Communicating transitive dependency vulnerabilities

A transitive dependency is the set of additional required components that get installed to make the root dependency function correctly. In the open source software (OSS) package manager ecosystem, an example of this is the installation of a single Pypi component that actually installs multiple other dependent components, demonstrated in the screenshot below.

Vulnerabilities for transitive dependencies are difficult for developers to manage, as they often require additional work on the developer to update a specific transitive dependency and the method for doing so differs from language to language. This work is often seen as costly on the

developer's end given the chance that introducing a newer version of a transitive dependency may introduce functional incompatibilities with the root package. The developer must also remember to remove the "override" to the specific transitive dependency once the root dependency updates to use the new version. Usually developers don't replace or maintain the dependencies in the components or packages that they use by themselves. In the example above, developers don't replace the needed `bleach` package dependencies like `webecodings` or `six` dependencies. In case of a vulnerability in these dependencies, it's the root package responsibility to update the necessary dependencies tree.

Several scanners can pick up transitive dependencies. For transitive dependencies, it is essential to assess whether your code leverages functionality from the transitive dependency. If you still need to, you may be able to exclude the package. Otherwise, if it does, you may need to work with the vendor's support organization or PSIRT to ensure the vulnerability has been reported and addressed. It is recommended that you harden your dependencies to reduce the attack surface by removing unused components, ports, or services and go with restrictive settings.

3.1.4 Selecting an SBOM format

In recent years, two SBOM formats have emerged as leading standards in the industry: Software Package Data Exchange (SPDX) and CycloneDX. Both formats have their own strengths and are widely used by organizations and communities across the world.

SPDX is an open standard that was created by the Linux Foundation's SPDX working group and is maintained by the Linux Foundation's Open Compliance Program. SPDX is also an ISO standard. SPDX provides a standard format for identifying software components and licenses and is supported by a large and growing community of contributors. The format is flexible, well-documented, and widely used, making it a popular choice for organizations looking to create and exchange SBOMs.

One of the key benefits of SPDX is its ability to provide a complete and accurate representation of software components and licenses. An SPDX SBOM can include detailed information about each component, including its name, version, and licensing information. This information helps vendors to track the components they use, understand their licenses, and ensure compliance with regulations.

CycloneDX is another widely used SBOM format that provides a simple and lightweight representation of software components. It focuses on the most critical information about software components and dependencies and was designed to be easy to use and understand. It

is a good choice for organizations who have limited resources, are just starting out with SBOMs, or are looking for a quick and straightforward solution.

Both CycloneDX and SPDX are compatible with a wide range of tools and systems. They can be used with SCA tools, vulnerability management platforms, and other systems that support SBOMs. Therefore, organizations can integrate either format into their existing workflows and processes with minimal disruption.

3.1.5 Making SBOMs available

Sharing SBOMs with customers can be done based on business needs or contractual obligations. However, it's important to treat the SBOM as a sensitive artifact, depending on the content, and only share it following appropriate access controls and safeguards. Some information, as an additional detail to the SBOM or an additional SBOM may be considered more sensitive than other information, such as information about corporate infrastructure, internal build processes and pipelines.

Some vendors and software providers have Non-Disclosure Agreements (NDAs) in place with their customers before sharing an SBOM. This ensures that sensitive information is not disclosed to unauthorized parties and that all parties involved understand the importance of protecting the SBOM's confidentiality.

Some producers may release SBOM information publicly or through controlled processes depending on the customer model and base and the level of information to their build processes. All SBOMs should include a hash uniquely identifying the SBOM or a full cryptographic signature to demonstrate the authenticity of the SBOM from the producer to avoid counterfeit SBOMs.

By carefully controlling access to the SBOM and having appropriate safeguards in place, companies can ensure that their customers have the information they need to make informed decisions about the software while also protecting their own intellectual property and sensitive information.

3.1.6 Updating SBOMs

The SBOM is initially created at a specific point in time and provides a static snapshot of the software's components at that moment. To maintain the accuracy and relevance of the SBOM, it is crucial to update it when changes are made to the software or its underlying components. If an update is made

to the software or its components, the SBOM should reflect the new state of the software. This ensures that the SBOM is always up to date and accurate.

When a software component is updated with a new build or release, a new SBOM must be created to reflect the updated version of the software. This allows for complete visibility into the software's components and helps to ensure that any vulnerabilities are identified and addressed in a timely manner.

Some producers may choose to release an SBOM for their major releases. Others may issue an SBOM for each major and minor release, and yet others for each code change of minor elements. Each of these approaches may be appropriate for their business and customer models. There is no universally accepted standard that all must follow the same approach.

3.2 VEX

A VEX document is a transitive file that associates a set of vulnerabilities to a particular set of components in an application or library that issues an SBOM. These vulnerabilities have been cataloged as CVEs and published publicly as part of a CNA's disclosure. A VEX document can also publish information regarding known exploitation against components. VEX shows the status of the vulnerabilities, whether there is or isn't a fix, the vulnerable condition, if the issue is still under investigation, and the severity rating for each CVE in relation to the component or product it has been issued for. A VEX may also indicate if the component or application has reached an End of Life (EOL) support condition from the upstream community or from the vendor producing the product whether software, hardware, or firmware. The VEX may also indicate if anything has been remediated and any potential effects from known exploitation.

Typically, a VEX is a transitive update that will be delivered via the Common Security Advisory Format (CSAF v2.x) file, usually in JSON format, with a software ID based on CPE or PURL to identify the relationship between a particular VEX and the SBOM. They could also be associated or discovered via a digital signature or hash or even through the use of a mechanism such as SIGSTORE.

The enormous and growing volume of potential vulnerabilities that can be detected makes it challenging for security teams to determine which vulnerabilities to prioritize and respond to. Many security tools can generate a large number of false positives, which can distract security teams from real vulnerabilities that threaten their organization and require their attention. This glut of information also makes it difficult for security teams to prioritize their remediation efforts based on exploitability and potential impact.

VEX can assist security teams to prioritize vulnerabilities based on criteria such as the likelihood of exploitation, the potential impact on critical systems or data, and the difficulty of remediation, and allow them to focus on the most critical issues and reduce the risk of a successful cyber attack.

A method for generating a VEX document / file:

1. Identify where your development team maintains their component inventory of all third-party components for a GA product.
    a. Development teams can generate a component inventory through software composition analysis tools or a supplier-provided SBOM.
2. Actively monitor the component inventory for vulnerabilities.
3. Assess the impact of the vulnerabilities following guidance defined in 3.2.5 Communicating Exploitability through VEX Documents, ensuring that the responses are customer appropriate.
4. Build a CSAF Generator by assessing how you map the source to the CSAF VEX Profile.
5. Generate the CSAF VEX profile, leveraging CSAF Generator and validator.
6. Publish the CSAF VEX profile per 3.2.3 Publication of VEX Advisories. Some examples of the publication of VEX in CSAF can be found from Cisco and from Red Hat.
7. Maintain the CSAF VEX profile by establishing triggers per 3.2.4 Processing SBOM Changes in VEX Documents.

VEX empowers security teams to perform:

Faster Response

Asset management systems automatically consume VEX advisories and process all relevant security information.

More Effective Remediation

With clear, up-to-date information about vulnerabilities and fixes, administrators can ensure that all necessary actions are taken.

Better Prioritization

Vendors reassess vulnerabilities in components based on how those components are used within the software and any additional protection mechanisms, resulting in more accurate dependency disclosures.

3.2.1 VEX use by customers

Vendors, software providers, and their customers can use VEX to get the latest published status of all vulnerabilities that are known to be exploited or to query a supplier to get the status of vulnerabilities in CISA's Known Exploited Vulnerability (KEV) Catalog.

3.2.2 Basic implementation for customer consumption

The following procedure outlines steps for basic implementation:

1. The customer determines the software version or versions that are deployed in their environment.
2. The customer partners with their PSIRT or vendor to find out where the VEX documents are published.
3. Check hash to ensure it has not been tampered with.
4. A revision history that provides a concise record of any changes, such as vulnerability status.
5. Build APIs or connect to the vendor's APIs.

6. Consume VEX documents that align with industry standards, such as CSAF.
7. Apply recommended vendor update, patch, or workaround.
8. For software versions that are under investigation or affected, the customers should follow up with the vendor on timelines and apply mitigations suggested by the vendor if they are available.

3.2.3 Publication of VEX advisories

VEX documents express the correlation between products, their components, and the vulnerabilities. VEX files can be distributed separately. To make customers aware of the known vulnerabilities, security teams can use VEX as a separately downloadable document. Having the VEX documents can help customers or engineers to do more accurate risk assessment and make decisions about which software to fix and prioritize.

VEX does not require SBOM, but can complement one if present.

VEX documents should be generated by software vendors and should contain all information about potentially affected products and components, including the fix status, utilizing the *under investigation* status when necessary. In VEX documents there can be references to the advisories, for example, in the default CSAF format, published by software vendors for fixed products or components. The VEX data should be publicly available for all customers.

Recommendations for publishing VEX documents:

- Use the product version download infrastructure to provide VEX documents.
- CSAF/VEX documents should be provided with the product version using its distribution infrastructure with the product_version ID for SBOM or CSAF used for their document association.
- Use version numbers and revision numbers per SBOM or CSAF conventions.
- Ensure that the VEX document is considered valid.
- Ensure that the VEX document is hashed to ensure the integrity of the content.

Note: VEX distribution via organizations that are not vendors are not addressed in this document.

3.2.4 Processing SBOM changes in VEX documents

Vulnerabilities related to a product can be published at any time, therefore, maintaining an accurate representation of exploits associated with the product is not trivial. Both vendors and customers must have systems in place to automatically detect new vulnerabilities associated with

a product. Establishing a proper expectation for updates to the VEX document is as important as updating them on a regular basis.

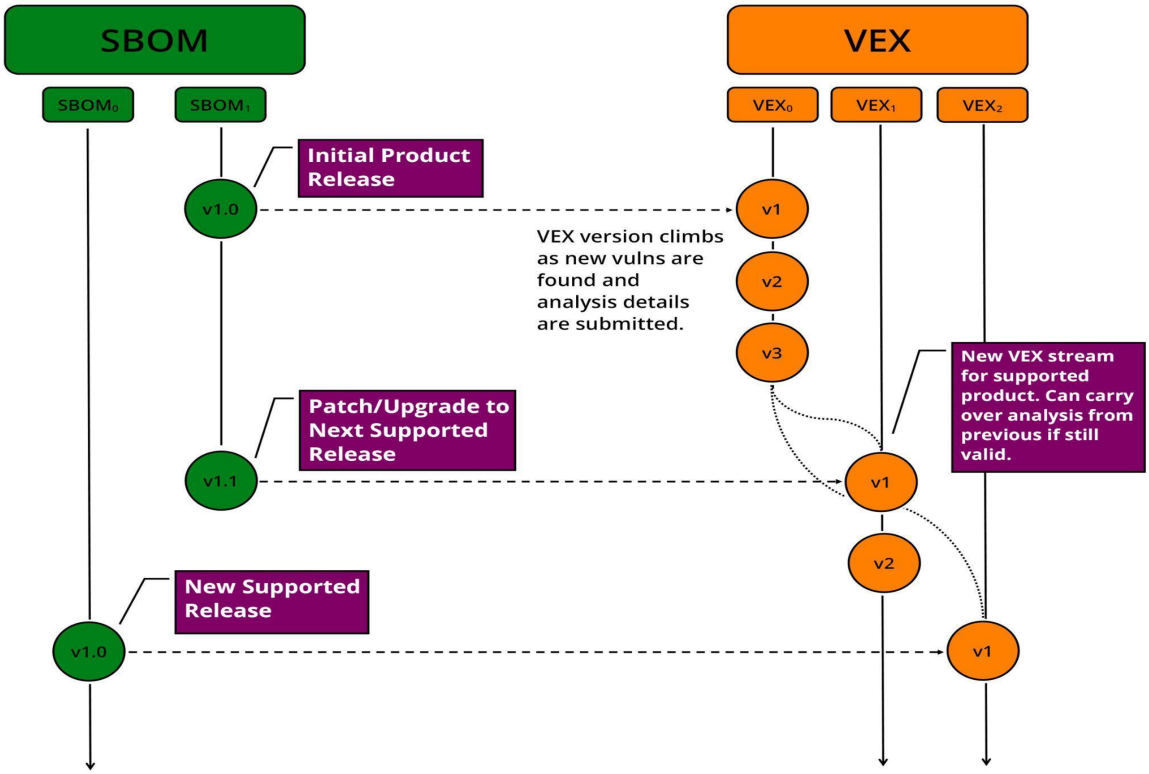
The SBOM lifecycle is closely related to the product release cycle. Each VEX document is associated with one or more specific product versions. Therefore, as a new product version and SBOM is released, all VEX documents that were associated with the previous product version will need to be updated or initialized to provide customers with applicable status. For example, when a vulnerability is resolved through a patch, mitigation, or if it is no longer present in the new release the VEX associated with, the previous release should now contain the upgrade instructions for customers. If the vulnerability is still present, a VEX may be updated to associate the new release version, or a new VEX may be initialized if your organization chooses to follow the single product version format option. All of this data should be stored in a way that these updates are not too complex to tackle. VEX information can then be queried as your organization is ready to publish it, or ideally, when the customer would like to ingest it.

The following vulnerability events may cause the VEX document to change without an update to the SBOM:

1. New vulnerabilities are published.
2. The details of a previously published vulnerability are updated.
3. The analysis state of impact and exploitability of a vulnerability changes.
4. The components affected change.

Note: Analysis for exploitability and mitigation efforts are a snapshot of one point in time, therefore, it is important to keep analysis data timestamped for comparison in cases 2, 3, and 4. Tools can be used to sync new and updated vulnerability information in your VEX; You will want to note when this requires reanalysis.

Diagram 3.2.4.1 The SBOM and VEX lifecycle



Consider Diagram 3.2.4.1 and imagine the following narrative. Acme Corporation is about to release version 1.0 of their new service. As part of this release, they created an SBOM ($SBOM_0$) for their customers. In doing so, a v1 VEX (VEX_0) document was also generated with audited justifications for various vulnerabilities found in the service. These three deliverables (software, $SBOM_0$, and VEX_0) makeup their release.

As time passes, new vulnerabilities are reported against dependencies in their service, resulting in the completion of further analysis. As a result, new VEX_0 versions can be made publicly available (v2, v3). Eventually, patches will be required due to new feature development or vulnerability fixes which materially affect the $SBOM_0$ content, and a new version of the software will be released (v1.1). $SBOM_0$ is updated to reflect the new service composition, and in the case of many cloud services, the previous $SBOM_0$ version (and service v1) is no longer in support.

Thus, VEX_0 v3 lifecycle will reach end of life (EOL), as does the v1 of the service and the associated $SBOM_0$ version. VEX1 v1 is generated against SBOM v1.1, and the team can optionally carry over their analysis results from VEX_0 v3 if they are still applicable. This new VEX_1 continues to change as analysis and vulnerabilities are updated (VEX_1 v2 onwards).

As Acme advances its service offerings, a new version of the service is launched with an associated SBOM ($SBOM_1$). V1.X ($SBOM_0$) customers are now considered legacy customers with

an EOL cycle of five years. Thus, as is with many organizations, Acme now supports two versions of their product with associated $SBOM_0$ and $SBOM_1$. As before, $SBOM_1$ has an associated VEX (VEX_3), which will change over time. Since this new service version is largely based on the same dependencies as the previous one, the team can utilize some past analysis from VEX_0 v3 into VEX_2 v1 where applicable.

In short, Acme only generates new publicly available SBOM versions when new releases are ready and maintains these documents until the product is EOL. The associated VEX documents are more dynamic in nature and change more rapidly but are always associated with at least one SBOM.

3.2.5 Communicating exploitability through VEX documents

We all know this story, a consumer of your product finds published vulnerabilities and reaches out, wanting to know why these issues are present. As a product security lead, you discuss with the product developers, and they assure you that this vulnerability is not exploitable (for one of many reasons); this is documented and communicated to the consumer. Then another consumer reaches out and you are in the same loop again.

This is the evolution of disclosure:

- Companies disclose vulnerabilities to the public via CVE. Data from CVE is absorbed in NVD, who often makes a separate evaluation based on general broad-based information.
- Scanning vendors largely leverage NVD information to identify vulnerable components with disclosed vulnerabilities. Not all aspects of a product can be scanned and therefore not all parts are disclosed. This often leads to significant challenges in determining the effects in products based on an NVD-induced interpretation.
- Companies provide SBOMS so you can accurately identify an ingredient list of components. However, the CSAF/VEX compliments SBOM by identifying the affected components from the ingredients based on use, protections, code updates, backports, or rebases. Hence, the advent of VEX will supersede the data from NVD (this will be a future issue for action).

SBOM is a necessary next step in software release and control. Still, the biggest fear for product security teams is that this will increase the number of vulnerabilities that consumers are requesting information. The VEX document(s) is the solution for this. Software vulnerabilities are a tricky business. We know that the same vulnerable component could be exploitable in one product, mitigated in another, and not affected in yet another. Differences in configurations add another layer of complexity.

The Cybersecurity and Infrastructure Security Agency (CISA), based on NTIA - Industry efforts, has published a list of required status options for each vulnerability listed in the VEX document.

- NOT AFFECTED – No remediation is required regarding this vulnerability.
- AFFECTED – Actions are recommended to remediate or address the vulnerability.
- FIXED – These product versions contain a fix for the vulnerability.
- UNDER INVESTIGATION – It is not yet known whether these product versions are affected by the vulnerability.

While details about the status applied to each vulnerability are only optional in the VEX document, the vendor must track these details with a relative timestamp for when analysis was completed or updated.

If the status is NOT AFFECTED, it is highly recommended you include one of the following status justifications:

- Component_not_present
- Vulnerable_code_not_present
- Vulnerable_code_cannot_be_controlled_by_adversary
- Vulnerable_code_not_in_execute_path
- Inline_mitigations_already_exist

Each of the status justifications is illustrated in detail in the Vulnerability Exploitability eXchange (VEX) - Status Justifications document.

You should seek legal guidance before publishing artifacts related to your software to identify any legal risks from the information they contain (for example., if a VEX document claims that a vulnerability is not exploitable but it later turns out to be, or if an SBOM contains information protected under an NDA with your suppliers).

VEX's implementation and use cases are flexible, and each organization may find that multiple use cases may apply depending on the situation.

See details on how to communicate these VEX justifications in the CylconeDX document.

Table 3.2.5 CISA Use cases supported by CycloneDX

Case #	Description	Support
Case-1	Single Product, Single Version, Single Vulnerability, Single Status	Supported

Case #	Description	Support
Case-2	Single Product, Single Version, Multiple Vulnerabilities, Single Status	Supported
Case-3	Single Product, Single Version, Multiple Vulnerabilities, Multiple Statuses	Supported
Case-4	Single Product, Multiple versions, Single Vulnerability, Single Status	Supported
Case-5	Single Product, All Versions, Single Vulnerability, Single Status	Supported
Case-6	Single Product, Multiple versions, Single Vulnerability, Multiple Statuses	Supported
Case-7	Multiple Products, Multiple Versions, Single Vulnerability, Multiple Statuses	Supported
Case-8	Multiple Products, Multiple Versions, Multiple Vulnerabilities, Multiple Statuses	Supported
Case-9	Multiple Product Lines, Multiple Vulnerabilities, Multiple Statuses	Not Supported

References:

1. ISO/IEC 29147:2018 Information technology — Security techniques — Vulnerability disclosure, https://www.iso.org/standard/72311.html
2. Goldstein, E. (2022, November 14). Transforming the Vulnerability Management Landscape. https://www.cisa.gov/news-events/news/transforming-vulnerability-management-landscape
3. Common Security Advisory Framework (CSAF). (n.d.). BSI. Retrieved May 26, 2023, from https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Industrielle-Steuerungs-und-Automatisierungssysteme/CSAF/CSAF_node.html