Minimum Requirements for Vulnerability Exploitability eXchange (VEX)

Publication date: April 2023

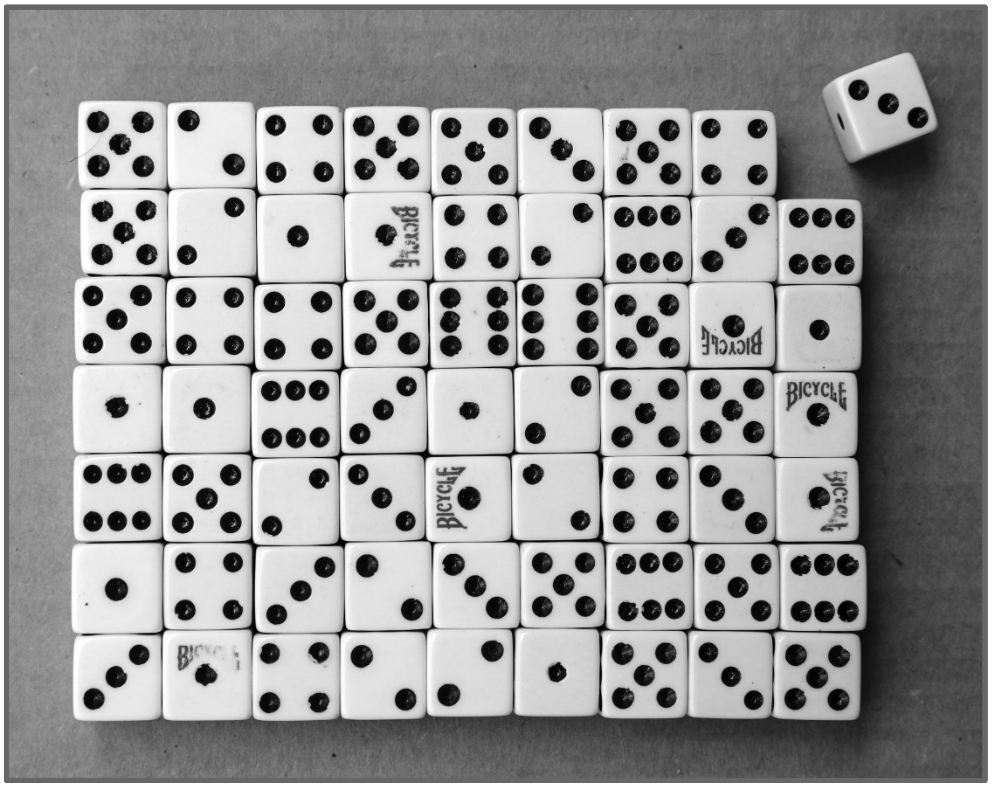

Photo by Mick Haupt on Unsplash

1.0 Overview

Vulnerability Exploitability eXchange (VEX) indicates the status of a software product or component with respect to a vulnerability.[1] A common VEX use case is to indicate that software is or is not affected by a vulnerability.

This document specifies the minimum elements to create a VEX document. These elements are derived from, but may not fully conform to, existing VEX documentation and implementations, as noted in section 4.1. This document also specifies some optional VEX elements.

At its core, a VEX data format is a machine-readable collection of information conveying the status of products or components with respect to a vulnerability.[2] As VEX has begun to be implemented across the diverse software ecosystem, the community sought to further define and specify the needed and optional elements to support automation, tooling, and interoperability.

VEX is designed to integrate with SBOM, vulnerability databases, and security advisories, but does not require any of these. VEX documents can be authored by the supplier of the software or by a third party.[3]

As practical implementation of VEX continues, changes to the minimum elements are expected. Elements may be added, removed, or changed, but the minimum requirements should allow for scalable implementations and should harmonize the community's expectations. Optional features can be harmonized as well.

1.1 About this document

This document defines the minimum elements independent of any format or implementation. This document builds on the open, international, community-led work around SBOM, and should not be read to re-specify or design SBOM or the definitions used in describing SBOMs.

This document was drafted and debated by experts from across the security and software world, representing different sectors and backgrounds. Participants wishing to be acknowledged are listed in section 4.3.

The drafting of this document did not follow a formal standards development process. This document does not necessarily represent consensus among broader communities around VEX, SBOM, and vulnerability management. This document does not represent official CISA policy, nor does the document impose or mandate compliance.

[1] While primarily designed for software vulnerabilities, VEX can convey status about cybersecurity vulnerabilities involving hardware, specifications, or other causes.
[2] For more information on VEX, including background and use cases, please see CISA.gov/SBOM and ntia.gov/SBOM.
[3] Terms such as "supplier" are used in accordance with existing SBOM terminology. See https://ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf.

This document follows the conventions defined in RFC 2119 (BCP14), primarily using MUST, SHOULD, and MAY.[4] Logically, any element that is not prohibited is implicitly allowed (MAY). This document uses MAY to note common or likely optional data elements. This use of MAY does not restrict the use of additional optional data elements that are not defined in this document.

All time and date elements MUST follow common international standards such as RFC 3339.[5]

Data element names are specified in [square_brackets]. When specified, data element values are specified in "quotes". Spaces in data element names and values are replaced_by_underscores. Data element names and values are case-insensitive.

2.0 VEX data elements

This version of the VEX data element requirements is 1.0.0. VEX implementations SHOULD declare which version of the requirements they support.

VEX does not specify, assume, or imply any default status that is not explicitly provided in VEX statements. VEX information may be incomplete. VEX does not require an author to provide a complete and comprehensive list of all products or components from a supplier or known to exist in the universe.

See Appendix A for a summary outline of VEX data elements.

2.1 VEX document

A VEX document is a container object holding one or more VEX statements (2.3).

A VEX document MUST contain at least one VEX statement.

A VEX document MUST provide required VEX document metadata and MAY provide other data.

The VEX data elements are organized around the concept of a document containing statements. VEX and VEX documents MAY be implemented within or as part of other formats or information systems. VEX information MAY be provided in whole or in part using services and APIs. Partial VEX information can be logically assembled into valid VEX documents and VEX statements. VEX information MAY be derived or synthesized from sources such as SBOMs, vulnerability management systems, security advisories, and software change management systems. For example, a Common Security Advisory Format (CSAF)[6] or CycloneDX[7] document identifier MAY be used as [doc_id]. Other formats or information systems that support VEX are

[4] https://rfc-editor.org/rfc/rfc2119 and https://www.rfc-editor.org/info/bcp14

[5] https://www.rfc-editor.org/rfc/rfc3339

[6] https://oasis-open.github.io/csaf-documentation/

[7] https://cyclonedx.org/

likely to have their own requirements. Figure 1 shows the high-level conceptual structure of the VEX document.

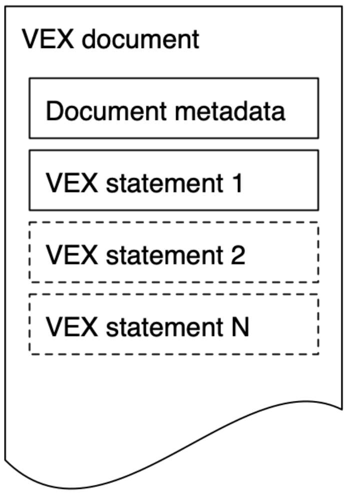

Figure 1: VEX document and statements

As of this writing, CSAF, CycloneDX, and OpenVEX[8] can contain or be used to generate VEX documents. Both CSAF and CycloneDX provide information beyond what is required for VEX.

2.2 Document metadata

To the greatest extent possible, VEX metadata is defined and maintained at the VEX document level. When appropriate and necessary, VEX metadata is defined at the VEX statement level. VEX document metadata MAY be synthesized or derived from VEX statement metadata; for example, [doc_time_last_updated] MUST be at least as recent as the newest [statement_time_last_updated]. VEX document metadata MUST accurately apply to all contained VEX statements.

2.2.1 Document ID [doc_id]

[doc_id] identifies a VEX document.

A VEX document MUST include one [doc_id].

[doc_id] SHOULD be managed within the [author] namespace, for example, [author]/[doc_id]. See A note on identity.

[doc_id] SHOULD be sufficiently unique with respect to context such as the [author] namespace and the time period during which the VEX document will be used.

[8] https://github.com/openvex

2.2.2 Document version [doc_version]

[doc_version] indicates the version of a VEX document.

A VEX document MUST include one [doc_version].

[doc_version] MUST be incremented when any content within the VEX document changes, including content in VEX statements contained within the VEX document.

[doc_version] MUST clearly convey positive incremental change.

2.2.3 Author [author]

[author] indicates the author of the VEX document. The [author] is responsible for the content of the VEX document.

A VEX document MUST identify the author.

[author] MUST be an individual or organization. To describe tools or other mechanisms used to generate VEX content, consider [tooling].

[author] MAY be a common name, or a URI.

[author] identity SHOULD be cryptographically associated with the signature of the VEX document or other exchange mechanism.

2.2.4 Author role [author_role]

[author_role] MAY specify the role of the [author].

[author_role] MAY use the “category of publisher” roles defined by CSAF 2.0: coordinator, discoverer, other, translator, user, vendor.[9]

2.2.5 Tooling [tooling]

[tooling] MAY specify tools or automated mechanisms that generate VEX documents, VEX statements, or other VEX information. Contrast with [author].

2.2.6 Timestamp first issued [doc_time_first_issued]

A VEX document MUST provide the date and time that the VEX document was first issued.

[doc_time_first_isued] MUST equal the oldest [statement_time_first_issued] of all included VEX statements.

[9] https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html#32181-document-property---publisher---category

2.2.7 Timestamp last updated [doc_time_last_updated]

A VEX document MUST provide the date and time that the VEX document was last modified.

[doc_time_last_updated] MUST initially be equivalent to [doc_time_first_issued].

[doc_time_last_updated] MUST reflect the most recently updated data in the VEX document. This means that [doc_time_last_updated] MUST be equal to or newer than the most recent [statement_time_last_updated], [impact_statement_time], or [action_statement_time] of all included VEX statements.

2.3 VEX statement

A VEX statement is a declaration that MUST convey a single [status] that applies to a single [vul_id] for one or more [product_id]s. Figure 2 shows the high-level conceptual structure of the VEX statement.

A VEX statement MUST be logically contained within a VEX document.

A VEX statement MUST exist only within one VEX document, that is, VEX statements are logically local to their containing VEX document.

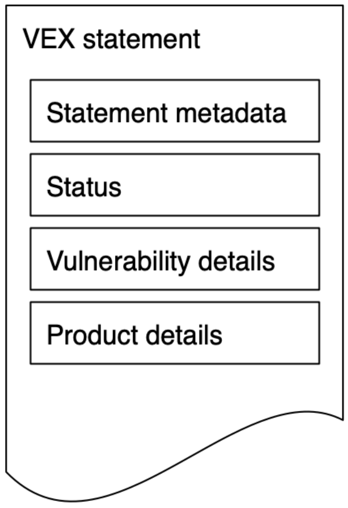

Figure 2: VEX statement

2.4 Statement metadata

To the extent possible, VEX metadata is stored in VEX documents. Certain metadata is specific to VEX statements.

2.4.1 Statement ID [statement_id]

[statement_id] uniquely identifies a VEX statement within a VEX document.

A VEX statement MUST be able to be specifically referenced within a VEX document.

A VEX statement SHOULD provide one [statement_id].

[statement_id] SHOULD be created within the [author] and [doc_id] namespaces and MAY be generated from other VEX information, for example, [author]/[doc_id]/[statement_id]. See A note on identity.

[statement_id] MAY minimally be an index of VEX statements within the scope of [doc_id].

2.4.2 Statement version [statement_version]

[statement_version] indicates the version of the VEX statement.

A VEX statement MUST provide one [statement_version].

[statement_version] MUST clearly convey positive incremental change.

[statement_version] MUST be incremented when any content within the VEX statement changes.

[statement_version] MAY be derived from or otherwise be related to [document_version].

2.4.3 Timestamp first issued [statement_time_first_issued]

A VEX statement MUST provide the date and time that the VEX statement was first issued.

[statement_time_first_issued] MAY be derived from or otherwise related to [doc_time_first_issued].

[statement_time_first_issued] MAY be derived from or otherwise related to [impact_statement_time] or [action_statement_time].

2.4.4 Timestamp last updated [statement_time_last_updated]

A VEX statement MUST provide the date and time that the VEX statement was last modified.

[statement_time_last_updated] MUST initially be equivalent to [statement_time_first_issued].

[statement_time_last_updated] MAY be derived from or otherwise related to [impact_statement_time] or [action_statement_time].

[statement_time_last_updated] MUST be equivalent to or newer than the most recent [impact_statement_time] or [action_statement_time].

2.5 Product details

Product details identifies and describes the products or components in a VEX statement.

Product details MUST include one or more [product_id] and MAY include one or more [subcomponent_id].

[product_id] and [subcomponent_id] SHOULD use existing and well-known identifiers.

[product_id and [subcomponent_id] SHOULD reference existing SBOM identifiers.

[product_id] and [subcomponent_id] SHOULD conform to reasonable and current conventions, for example, follow a "supplier/product/version" construct.

[product_id] and [subcomponent_id] MAY be URIs, URLs, hashes, commit IDs, versions, version ranges, dates, date ranges, or any other identification system.

[product_id] and [subcomponent_id] MAY be arbitrarily created by the [author].

[product_id] and [subcomponent_id] MAY specify sets of products or components, for example:

- Every product or component owned by a supplier
- A product family or product line
- Version ranges
- A specific branch

See also VEX references.

2.5.1 Product identifier [product_id]

[product_id] MUST identify the product or component that [vul_id] and [status] applies to.

[product_id] MAY specify a set of products or components and MUST specify at least one of:

- [subcomponent_id]
- A component (often a sub-component of a product)
- A product, for example, a final good assembled
- A set of products or components, for example, a product line or family
- A supplier (indicating the set of all products or components from the supplier)

The terms "product" and "component" are further explained in section 3.6.1.

2.5.2 Subcomponent identifier [subcomponent_id]

A VEX statement MAY include one or more identifiers for subcomponents associated with vulnerability details.

A VEX statement asserts the [status] of [product_id] with respect to [vul_id]. A VEX statement MAY also convey that [subcomponent_id] is included in [product_id]. A common VEX use case is to convey that [subcomponent_id] is "affected" by [vul_id] while [product_id] is "not_affected" by [vul_id].

[subcomponent_id] MAY be derived from [product_id], particularly if [product_id] is associated with SBOM or other references that convey dependencies.

[subcomponent_id] MAY be derived from [vul_id] or [vul_description].

2.5.3 Supplier [supplier]

Product details SHOULD identify the [supplier] of [product_id] or [subcomponent_id].

[supplier] MUST clearly indicate the [product_id] or [subcomponent_id] to which [supplier] applies. For example:

[supplier]/[product_id]

[supplier]/[subcomponent_id]

2.6 Vulnerability details

Vulnerability details identify and provide information about the vulnerability in a VEX statement.

See also VEX references.

2.6.1 Vulnerability identifier [vul_id]

[vul_id] identifies the vulnerability in a VEX statement.

A VEX statement MUST specify one [vul_id].

[vul_id] SHOULD use existing, readily available, and well-known identifiers such as: CVE,[10] the Global Security Database (GSD),[11] or a supplier's vulnerability identification system. It is expected that vulnerability identification systems are external to and maintained separately from VEX.

[vul_id] MAY be URIs or URLs.

[vul_id] MAY be arbitrary and MAY be created by the [author].

2.6.2 Description [vul_description]

A VEX statement MUST include or reference one [vul_description] that corresponds to [vul_id].

[vul_description] MUST either be included in the VEX statement or made available to VEX consumers (for example, through a URL).

[10] https://www.cve.org/
[11] https://globalsecuritydatabase.org/

2.7 Status

2.7.1 Status [status]

A VEX statement MUST provide one [status] that applies to all contained [product_id]s with respect to [vul_id].

[status] MUST be one of the following values, some of which have further requirements:

- Not affected (“not_affected”)
- Affected (“affected”)
- Fixed (“fixed”)
- Under investigation (“under_investigation”)

2.7.1.1 Not affected (“not_affected”)

No remediation or mitigation is required. The vulnerability does not affect the listed [product_id]s.

2.7.1.1.1 Impact statement [impact_statement]

For [status] “not_affected”, if [justification] is not provided, then a VEX statement MUST provide an [impact_statement] that further explains how or why the listed [product_id]s are “not_affected” by [vul_id].

If [justification] is provided, then a VEX statement MAY provide an [impact_statement].

2.7.1.1.2 Timestamp of impact statement [impact_statement_time]

[impact_statement] MAY include [impact_statement_time], recording when the [impact_statement] was issued.

2.7.1.1.3 Justification [justification]

For [status] “not_affected”, a VEX statement SHOULD provide [justification].

If [justification] is not provided then [impact_statement] MUST be provided.

[justification] MUST be one of the following values, described further in the previous publication, *Vulnerability Exploitability eXchange (VEX) - Status Justifications*.[12]

- “Component_not_present”
- “Vulnerable_code_not_present”
- “Vulnerable_code_not_in_execute_path”
- “Vulnerable_code_cannot_be_controlled_by_adversary”
- “Inline_mitigations_already_exist”

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

2.7.1.1.3.1 *“Component_not_present”*

The vulnerable [subcomponent_id] is not included in [product_id].

2.7.1.1.3.2 *“Vulnerable_code_not_present”*

The vulnerable [subcomponent_id] is included in [product_id] but the vulnerable code is not present. Typically, this case occurs when source code is configured or built in a way that excludes the vulnerable code.

2.7.1.1.3.3 *“Vulnerable_code_not_in_execute_path”*

The vulnerable code (likely in [subcomponent_id]) cannot be executed due to the way it is used by [product_id]. Typically, this case occurs when [product_id] includes the vulnerable code but does not call or otherwise use it.

2.7.1.1.3.4 *“Vulnerable_code_cannot_be_controlled_by_adversary”*

The vulnerable code is present and used by [product_id] but cannot be controlled by an attacker to exploit the vulnerability.

2.7.1.1.3.5 *“Inline_mitigations_already_exist”*

[product_id] includes built-in protections or features that prevent exploitation of the vulnerability. These built-in protections cannot be subverted by the attacker and cannot be configured or disabled by the user. These mitigations completely prevent exploitation based on known attack vectors.

2.7.1.2 Affected (“affected”)

Actions are recommended by [author] to remediate, mitigate, or otherwise address [vul_id]. The vulnerability affects the listed [product_id]s.

2.7.1.2.1 Action statement [action_statement]

For status “affected”, a VEX statement MUST include one [action_statement] that SHOULD describe actions to remediate or mitigate [vul_id].

2.7.1.2.2 Timestamp of action statement [action_statement_time]

[action_statement] MAY include [action_statement_time] recording when the [action_statement] was issued.

2.7.1.3 Fixed (“fixed”)

The listed [product_id]s contain fixes for [vul_id].

2.7.1.4 Under investigation (“under_investigation”)

The [author] of the VEX statement or other relevant parties are investigating and have not yet declared a final [status].

It is expected that [status] “under_investigation” will change once the investigation has reached a conclusion.

2.7.2 Status notes [status_notes]

[status_notes] MAY convey information about how [status] was determined and MAY reference other VEX information.

3.0 Usage

The primary purpose of this document is to define VEX data elements. This section provides some additional guidance on using VEX.

3.1 VEX references

VEX statements identify, refer to, or define products (components, subcomponents) and vulnerabilities.

3.1.1 External references

It is expected, but not required, that VEX product details and vulnerability details reference external data sources. It is expected that these identification systems are external to and maintained separately from VEX.

Product details (2.5) specifies the products or components to which [status] applies. Product details SHOULD reference existing SBOM identifiers.

Vulnerability details (2.6) specifies one vulnerability per VEX statement.

Product details and vulnerability details SHOULD use existing and well-known identifiers.

Product details and vulnerability details MAY be arbitrary. Arbitrary product and vulnerability details SHOULD conform to reasonable and current conventions, for example, product details SHOULD follow a "supplier/product/version" construct, and vulnerability details SHOULD use an existing vulnerability identification system.

VEX information SHOULD facilitate automation. To do so, VEX data MAY be typed, that is, a VEX implementation MAY declare the type of data elements or references.

3.1.2 Multiple references

A VEX statement MUST identify at least one product (or component) and exactly one vulnerability.

A VEX statement MAY reference more than one product as long as [status], [vul_id], and other VEX information are correct for the complete set of products. If status or other VEX information changes for a subset of products, additional VEX statements MUST be created for the respective subset.

This document does not specify how to define sets of products or components, nor does VEX specify how to define version ranges. The reader should look to SBOM or VEX implementations for this.

To issue multiple VEX statements, an [author] MAY issue one VEX document containing multiple statements or multiple VEX documents each containing one or more VEX statements.

Use cases involving multiple products and vulnerabilities are defined in the previous publication, *Vulnerability Exploitability eXchange (VEX) – Use Cases.*[13]

3.2 Metadata inheritance

VEX documents (and included VEX statements) MUST be able to exist without additional information infrastructure, that is, a VEX document does not have to be (part of) a vulnerability advisory.

A VEX statement MAY inherit, or depend on, metadata from a containing VEX document, such as a vulnerability advisory (for example, CSAF, CycloneDX). In such a case, the advisory MUST provide VEX metadata needed by VEX statements.

VEX documents and included VEX statements MUST maintain independent metadata when necessary, for example, if document metadata such as [tooling] or [author_role] changes, [doc_time_last_updated] and [doc_version] would be different (more recent) than any included [statement_time_last_updated] and [statement_version]. Similarly, if one of many VEX statements within a VEX document changes, [doc_time_last_updated] and [doc_version] MUST be updated to reflect the change.

If a VEX statement is detached from its original VEX document, a new VEX document MUST be created and the new VEX document MUST copy (inherit) appropriate document metadata from the original VEX document.

3.3 Cryptography

VEX implementations MUST support commonly accepted and current digital signature and encryption mechanisms. Cryptography MAY be applied at the VEX statement or VEX document level. Cryptography MAY be applied externally, for example, using a detached signature, capabilities of a containing VEX document, or at a transport level such as HTTPS.

Exchanging VEX documents over HTTPS MAY be sufficient, assuming the VEX consumer sufficiently trusts and understands the certificate chain, identity of the HTTPS provider, and identity of the VEX [author].

VEX documents and statements SHOULD be signed. VEX documents and statements MAY be encrypted. VEX documents SHOULD cryptographically associate [author] with the identity of the signer.

The signature or encryption of a VEX document MUST cover all the document metadata and all the VEX statements within the document. A VEX statement MAY rely on cryptography provided by the containing VEX document.

[13] https://www.cisa.gov/sites/default/files/publications/VEX_Use_Cases_Apr22.pdf

3.4 A note on exchange

This document does not specify how or when to exchange or track changes to VEX information. Timestamps and versions of VEX documents, VEX statements, and other elements are intended to support change notification and tracking.

As noted in the VEX document (2.1) section, VEX data elements are organized around the concept of a document containing one or more statements. VEX information MAY be provided in whole or in part using services and APIs. Partial VEX information can be logically assembled into valid VEX documents and VEX statements. VEX information MAY be derived or synthesized from sources such as SBOMs, vulnerability management systems, security advisories, and software change management systems.

VEX MUST support cryptography (3.3), which MAY be used to control exchange.

It is expected that VEX statements will often accompany vulnerability information, for example, a vulnerability advisory could also be a valid VEX document (or include VEX documents).

3.5 A note on identity

It is important to have authentic (and accurate) VEX information. This largely relies on the identity of the VEX [author].

VEX authors SHOULD identify themselves cryptographically using digital signatures and the [author] field.

VEX statements SHOULD use the author's identity as a namespace, that is, [author] SHOULD be used as a namespace partition. Assuming no collisions in [author], the author is free to determine their own [doc_id] and [statement_id] values.

VEX document and statement identifiers SHOULD follow this convention:

> [author]/[doc_id]
>
> [author]/[doc_id]/[statement_id]

3.6 Limited glossary

This document uses the terms defined in Section 4 of *Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)* with the following modifications.[14]

[14] https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf.

3.6.1 Product

The term “product” as used in this document means “a unit of software as defined by a supplier or author.” “Unit of software” can be understood to also include software and cyber-physical systems, devices, specifications, and hardware. The term “product” is equivalent to the SBOM framing term “primary component.”

4.0 Acknowledgements

4.1 Existing sources

The VEX requirements are significantly influenced by, but do not necessarily fully conform to, the following sources.

- Vulnerability Exploitability eXchange (VEX) – Use Cases[15]
- Vulnerability Exploitability eXchange (VEX) - Status Justifications[16]
- CSAF VEX profile
  - https://docs.oasis-open.org/csaf/csaf/v2.0/os/csaf-v2.0-os.html#45-profile-5-vex[17]
  - https://docs.oasis-open.org/csaf/csaf/v2.0/os/schemas/csaf_json_schema.json[18]
- CycloneDX
  - CycloneDX - Vulnerability Exploitability eXchange (VEX)[19]
  - https://github.com/CycloneDX/bom-examples/tree/master/VEX[20]
- Healthcare SBOM Proof of Concept[21] (ongoing)

4.2 Document feedback

If you have any feedback on the contents of this paper, please send us your thoughts at SBOM@cisa.dhs.gov. Your feedback will be valuable to us in making continual improvements to the paper. VEX documents are starting to be used across the software ecosystem, and the concept will continue to be refined as implementers and adopters encounter new challenges and opportunities. Furthermore, this document does not contain an exhaustive list of status justifications, and future work may contain new status justifications as demand arises.

4.3 Participants

This document was a product of the VEX Working Group, which grew out of the NTIA Multistakeholder Process and the Framing Working Group, initially beginning work in 2020. That work continued into 2023, facilitated by CISA.

Participants included:

Adolfo García Veytia, Chainguard
Ali Fessi, Robert Bosch GmbH

[15] https://www.cisa.gov/sites/default/files/publications/VEX_Use_Cases_Aprill2022.pdf
[16] https://www.cisa.gov/sites/default/files/publications/VEX_Status_Justification_Jun22.pdf
[17] https://docs.oasis-open.org/csaf/csaf/v2.0/os/csaf-v2.0-os.html#45-profile-5-vex
[18] https://docs.oasis-open.org/csaf/csaf/v2.0/os/schemas/csaf_json_schema.json
[19] https://cyclonedx.org/capabilities/vex/
[20] https://github.com/CycloneDX/bom-examples/tree/master/VEX
[21] An early stage of the Healthcare SBOM Proof of Concept work is documented in https://ntia.gov/sites/default/files/publications/ntia_sbom_healthcare_poc_report_2019_1001_0.pdf.

Allan Friedman, CISA
Art Manion, ANALYGENCE Labs
Aruneesh Salhotra, Nomura
Bob Haack, Johnson & Johnson MedTech
Brandon Lum, Google
Bruce Lowenthal, Oracle Corporation
Bryan Cowan, Fortress Information Security
Cassie Crossley, Schneider Electric
Charles Long, Arthrex, Inc.
Charlie Hart, Hitachi, Ltd.
Charlie Jones, ReversingLabs
Christine O'Leary, Intel
Christopher Hibbard, Hewlett Packard Enterprise
Curtis Yanko, Grammatech
Dan Luhring, Chainguard
Daniel Bardenstein, Manifest Cyber
Deanna Medina, Honeywell
Derek Kruszewski, aDolus Technology Inc.
Duncan Sparrell, sFractal Consulting
Ed Heierman, Abbott
Hendrik Tjoelker, Hanze University Groningen
Ixchel Ruiz, Jfrog
Jeremiah Stoddard, INL
Jim Jacobson, Siemens Healthineers
Jonathan Spring, CISA
Jorge Acevedo Canabal, MD, Biohacking Village
Josh Bressers, Anchore
Joyabrata Ghosh, Elektrobit
Justin Murphy, CISA
Kosta Kalpos, Splunk
Kuldeep Sandhu, CISA
Larry Feldman, Ph.D., HII
Megan Doscher, CISA
Mehdi Mirakhorli, Rochester Institute of Technology (RIT)
Mike Powers, Intermountain Health
Nadeem Anwar, AT&T
Paavaanan Tamil Iraivan, Gigamon Solutions Private Ltd.
René Pluis, Philips
Ricardo A. Reyes, Tidelift, Inc
Rich Steenwyk, GE HealthCare
Samuel Moore, T-Mobile
Sandeep Patil, Philips
Scott Armstrong, Interos.ai
Tania Ward, Dell

Thomas Schmidt, Federal Office for Information Security (BSI) Germany
Ty Greenhalgh, Claroty
Yotam Perkal, Rezilion
Zvika Ronen, FOSSAware

Annex A: Index of VEX data elements

This index summarizes the structure of VEX data elements and values.