# Secvisogram Author Guide

BSI

2022-08-01

## Table of Contents

# 1 Document

Contains all metadata that is required to identify the advisory.

This includes, among other things, the title and category (profile) of the document, as well as the publisher and version history.

A machine-readable advisory requires the same - or more - information than the human-readable version.

If more information is provided than in the human-readable format, this should be noted for the review of the document.

The human-readable advisory must be derivable from the machine-readable one. The machine-readable document is superordinated to the human-readable document.

**Category**

This value determines the profile name for the CSAF document.

It should always be the value of the profile defined in the specification.

The selected profile has an impact on how the document is validated.

It is highly encouraged to use the defined CSAF profiles.

The available profiles are:

- `csaf_base`
- `csaf_informational_advisory`
- `csaf_security_incident_response`
- `csaf_security_advisory`
- `csaf_vex`

As advisories with category `csaf_base` do not provide much value, it is recommended to use a higher profile.

For validation, the profile `csaf_base` is always used as fallback. With this profile, fewer validations are carried out.

In general, if in doubt, use a higher profile containing more information.

A company may introduce custom categories, which will also lead to `csaf_base` being used as fallback for validation.

For a document with the profile `csaf_base` the category value must not be equal to the (case insensitive) name (without the prefix `csaf_`) or value of any other profile than “CSAF Base”.

Any occurrences of dash, whitespace, and underscore characters are removed from the values on both sides before the match.

Also, the value must not start with the reserved prefix `csaf_` except if the value is `csaf_base`.

**Csaf Version**

Must be set to `2.0` which is currently the only valid value.

**Lang**

The language the document is written in.

Must not be the same as `/document/source_lang`.

Helps the consumer to automatically filter the documents according to his language.

Can also be used to sort the documents by language within the distribution mechanism (see section 7 of the specification).

As part of the document creation process, the language helps to run a spell checker.

Therefore, it should be as specific as possible and reasonable: e.g. use `en-US` or `en-GB` instead of just `en`.

Further description of this type can be found under types.

**Source Lang**

This field must be present and set if the value `translator` is used in `/document/publisher/categoy` to indicate the language of the original document.

If the document was not translated, this field should not be set.

The source language must not be the same as the document language in `/document/lang`.

Further description of this type can be found under types.

**Title**

The title should provide a quick overview of what this document is about to the user.

Typically, a product name is referenced. For vulnerabilities that became more well known,

several exemplary products may be listed.

*Examples:*

- `Product A affected by RCE`
- `Multiple Products affected by Log4Shell Vulnerability`
- `Product Family X affected by Vulnerability Y`

## 1.1 Acknowledgments

Acknowledgments of people and organizations that were involved in discovery, publication and coordination of the process that lead to the CSAF document.

The acknowledgments are an important tool to reward and honor security researchers and contribute to the reputation of an organization.

In contrast to the Vulnerability Acknowledgments (`/vulnerabilities[]/acknowledgments`) contributions to the whole document can be honored.

Listing coordinating entities is recommended but not required.

Further description of this type can be found under types.

## 1.2 Aggregate Severity

Vendor assessment of the overall criticality of the vulnerabilities in the document.

This could, for example, be the textual description of the base severity corresponding to the highest CVSS score.

**Namespace**

Reference to an explanation of the meaning of the severity rating.

**Text**

Represents overall assessment and should be a string from a closed set.

The set may be vendor-specific or come from an applicable standard.

*Examples:*

- SSVC decision or vector
- Red Hat Severity Ratings
- Assessment according to the German Advisory Format (DAF)
- Rating between 1 and 5

## 1.3 Distribution

Specifies how and to whom the document can be distributed.

Classification according to TLP is preferred because it is machine-processable.

**Text**

A textual description of how the document may be distributed.

The use of TLP labels is preferred.

### 1.3.1 TLP

The Traffic Light Protocol (TLP) is a standardized agreement for the exchange of information worthy of protection

(see https://www.first.org/tlp/docs/tlp-v1.pdf).

Public Advisories should generally be classified as `TLP:WHITE`.

If there are prior obligations to inform a closed user group, these should be distributed as `TLP:AMBER` or `TLP:RED`.

In principle, the definition of FIRST applies.

If a different definition has been agreed with the customer group, this must be specified with a URL (`/document/distribution/tlp/url`).

**Label**

The label should be set and must be valid. There are four traffic light values:

- `RED` - *Not for disclosure, restricted to participants only*:

  In the context of a meeting, for example, `TLP:RED` information is limited to those present at the meeting.

  The distribution of `TLP:RED` information will generally be via a defined list and in some circumstances may only be passed verbally or in person.

- `AMBER` - *Limited disclosure, restricted to participants' organizations*:

  The recipient may share `TLP:AMBER` information with others within their organization, but only on a 'need-to-know' basis.

  The originator may be expected to specify the intended limits of that sharing.

- `GREEN` - *Limited disclosure, restricted to the community*:

  Information in this category can be circulated widely within a particular community.

  However, the information may not be published or posted publicly on the Internet, nor released outside the community.

- `WHITE` - *Disclosure is not limited*:

  Subject to standard copyright rules, `TLP:WHITE` information may be distributed freely, without restriction.

For more information about the TLP, refer to the TLP website from FIRST.

**Url**

Reference to the meaning of the used TLP labels.

The default value is a link to the definitions from FIRST.

This default value should not be omitted.

The default value should only be deviated from in exceptional cases, since the default is easier to automate.

## 1.4 Notes

For CSAF advisories that contain a `product_tree`, a short description of the products should be included.

This should have the category `description` and the title “Product Description”.

In this note, the product should be described in more detail in terms of functionality and benefits in a few sentences.

This applies analogously to product families.

Another element with title “General Security Recommendations” and category `general` should be created.

Another element of the category `summary` that summarizes vulnerabilities should be created.

Adding a legal disclaimer for the document is encouraged.

FAQs, if not vulnerability specific, belong in this section.

Further description of this type can be found under types.

*Additional details for profile CSAF Informational Advisory*

For the `csaf_informational_advisory` profile at least one note must exist which has a `category` of `description`, `details`, `general` or `summary`.

It should contain information what the “issue” is about and describe it as well as countermeasures or recommended actions.

*Additional details for profile CSAF Security Incident Response*

For the `csaf_security_incident_response` profile at least one note must exist which has a `category` of `description`, `details`, `general` or `summary`.

The response should be detailed. It may touch on impacts and recommended actions.

The response should be clearly marked as such - e.g. through the title of the note.

## 1.5 Publisher

Contains all information about the issuing party of the CSAF document.

**Category**

The publisher’s category is useful for the reader to understand the perspective of the publisher.

This information enables the consumer to estimate the trustworthiness of the advisory.

The valid values are

- `coordinator`
- `discoverer`
- `other`
- `translator`
- `user`
- `vendor`

In most cases the value `vendor` is correct.

If the value `translator` is used, the field `/document/source_lang` must be set.

**Contact Details**

Lists options of contacting the publisher of the advisories.

At least one email address should be given for queries about the CSAF document.

Email addresses of functions are preferred since people may change but the functional mailbox remains.

If in doubt, a website with a contact form can also be given.

For quick queries, a telephone number can also be provided.

Listing postal addresses is unusual but possible.

It can be useful, especially if a telephone number is given, as it helps to get an idea what timezone the team is located in.

**Issuing Authority**

Used to define responsibilities.

Can also be used to establish geographical responsibility, e.g. Cert only for national security researchers and/or vendors.

It is mostly used for sectoral, geographical or organizational specialization.

*Example:*

- ICS-CERT may state that “[they] coordinate any ICS related vulnerabilities”.

  This means that ICS-CERT is not the right team to talk with in regards to IT products (e.g. Microsoft Exchange).

Especially relevant for large companies/organizations or managed service providers.

Also relevant if there are several teams that issue advisories within the same group.

*Example:*

- Despite the “similar” name of “Siemens” and “Siemens Energy”, responsibilities can be described more clearly.

The value should be a full text description.

*Example:*

- “Example Company PSIRT is responsible for any vulnerabilities related to Example Company’s products or services.”

**Name**

The name of the issuing party. Typically, the company name or the name of the Product Security Team.

*Examples:*

- `Example Company PSIRT` (Product Security Incident Response Team)
- `ProductCERT` (Product Computer Emergency Response Team)
- `Example Company`

**Namespace**

The namespace is useful to unambiguously and globally identify the vendor or manufacturer.

Often times a URL of the company or issuing party is used which is recommended.

Can also be a URL pointing to a page on which advisories are published.

In combination with the tracking ID (`/document/tracking/id`) the CSAF document is uniquely identifiable.

If the document version is added to that combination, a single version of the document is globally uniquely identified.

## 1.6 References

There should be one reference item that provides a canonical URL for this document:

It has the category `self`, the `url` starts with `https://` and ends with a valid filename for the CSAF document according to the rules in section 5.1. of the specification.

All links that have any relevance in the overall context of the document should be listed here.

This could include original advisories, advisories of the manufacturer or advisories of the coordinator.

If other advisories (especially from suppliers) are referenced, they should be listed here.

It is recommended to include links to the recommended cybersecurity best practices documents of the organization and the website which lists the advisories.

CVE entries should only be linked in the corresponding vulnerability item.

Further description of this type can be found under types.

*Additional details for profile CSAF Informational Advisory*

For the `csaf_informational_advisory` profile at least one reference must exist which has links to an `external` source.

The sources linked in such external references may be documents or websites providing more details about the issue or its remediation (if possible).

This could be a hardening guide, a manual, best practices or any other helpful information.

*Additional details for profile CSAF Security Incident Response*

For the `csaf_security_incident_response` profile at least one reference must exist which has links to an `external` source.

The sources linked in such external references may be documents or websites providing more details about the incident.

## 1.7 Tracking

Provides all meta information required for tracking the document and its lifecycle.

This includes the change history and the current state.

**Current Release Date**

Defines when the current version of the document was published.

Should not be older than the `date` of the newest item in Revision History (`/document/tracking/revision_history`).

The time should reference the point in time of the actual (planned) publication.

**ID**

The unique name of the document within the issuing party.

In combination with the publisher namespace (`/document/publisher/namespace`) the CSAF document is globally identifiable.

If the document version is added to that combination, a single version of the document is globally uniquely identified.

The tracking ID remains constant when a new version of the document is released.

The filename of the document is derived from this ID as described in the specification.

The ID is typically a combination of a short identifier of the issuing party, year and a sequential number.

The recommended format is “CompanyAbbreviation-YYYY-#####”

*Examples:*

- `ECSA-2022-00123` (Example Company Security Advisory)

**Initial Release Date**

The date when the first version of the document was published.

In pre-release versions the planned publication date is used.

The value should not be older than the `date` of the oldest item in Revision History (`/document/tracking/revision_history`).

Typically, this is identical to the oldest element in the revision history.

For published documents, the oldest item has the `number` value of `1` for integer versioning

and `1.0.0` for semantic versioning.

The value of initial release date does not change anymore after publication of that first version.

The time should reference the point in time of the actual (planned) publication.

**Status**

The working status of the document. Possible values are `draft`, `interim` and `final`.

Must be set to `draft` for all draft versions, i.e. whenever the version is `0` or `0.x.y`, or contains a pre-release part.

If it is foreseeable that updates will come in high frequency within the next days, then the value `interim` should be selected.

The status `interim` asks the consumer to check the advisory more often for updates.

This is especially the case for vulnerabilities where there are frequent updates while the analysis is still running (e.g. like Log4Shell).

The value `final` should be chosen for all versions where no exceptionally high frequency of updates is expected.

For example, if advisories are released once a month on a regular patch day, the value `final` can be specified.

If it is generally not foreseeable that changes will be made at the time of release, the value `final` should be selected.

**Version**

The version of the present document.

The version may still contain pre-release information (e.g. 1.0.0-17) or build metadata for documents in `draft` status.

For documents in `interim` or `final` status no pre-release information must be included.

In general, a distinction is made between integer versioning and semantic versioning.

With integer versioning the version number is incremented by one for every public release.

With semantic versioning the version number has the form `MAJOR.MINOR.PATCH` and the individual parts are incremented under certain conditions.

Whenever the operator needs to do a new matching run on his asset database (matching the products from the CSAF product tree with deployed products) the `MAJOR` version is incremented.

For small changes such as correcting typos, the `PATCH` version is incremented.

All changes in between trigger an increase of the `MINOR` version.

The exact rules of the versioning schemes are defined in section 3.1.11 of the specification.

Further description of this type can be found under types.

### 1.7.1 Aliases

Aliases can be used to indicate the number, identifier or name assigned to this advisory, case or issue by the coordinator, manufacturer, supplier or other party.

This can also include internal ticket numbers.

The aliases should not be used to list identifiers of single vulnerabilities which are listed in `/vulnerabilities[]/ids`.

*Examples:*

- `ICSA-20-168-01`
- `VU#257161`
- `Ripple20`
- `AMNESIA:33`
- `OT:ICEFALL`

**Alias**

An alias name.

### 1.7.2 Generator

The generator that was used to generate the CSAF document.

**Date**

The date when the document was generated.

May be different from the `current_release_date`.

#### *1.7.2.1 Engine*

Filled in by the generating program.

**Name**

Filled in by the generating program, represents the name of the program.

*Examples:*

- `Red Hat SDEngine`
- `Secvisogram`
- `TVCE`

**Version**

Different versions of the generator may generate CSAF documents with potentially different structure.

Based on this value, different paths may need to be used for matching.

*Examples:*

- A company may change their name and in consequence `full_product_names` and the generation of document changes.

  This would require an adjustment of matching.

A change in versioning (e.g. switching from integer versioning to

semantic versioning) requires a change in matching.

*Examples:*

- `0.6.0`
- `1.0.0-beta+exp.sha.a1c44f85`
- `2`

### 1.7.3 Revision History

The revision history is used to track the development of the document in at least all published versions.

There must be an entry in the revision history for each published version.

For pre-release versions there are no entries in the version history. However, before the first release, the field can be used to record changes.

The revision items sorted by date must have `number` values in ascending order.

The `number` of the revision item with the most recent `date` value must be the same as in `/document/tracking/version`.

For this comparison build metadata is ignored.

If the document status is `draft` any pre-release part is also ignored.

The revision items sorted by date must not omit a version `number`.

In case of semantic versioning,

this applies only to the major version.

This sorted list of version numbers must start with (major) version `0` for documents in `draft` status or `1` in `final` or `interim` status.

The same version `number` must not be repeated for multiple revision items.

The oldest revision item should not have a newer `date` than the initial release date `/document/tracking/initial_release_date`.

#### *1.7.3.1 Revision History*

An item in the revision history.

**Date**

The date when this version was published. For pre-release versions the planned publication date is used.

For final and interim versions the actual publication date is used.

The value does not change anymore after publication.

*Example:*

- The publication of an advisory was planned on 18.05. at 12:00.

  If the advisory is then published at 11:30, the time should be adjusted.

If the time is not corrected before publication, it must not be changed afterwards anymore.

**Legacy Version**

The legacy version can be used to create a mapping to a human-readable version of the document that follows a version scheme other than

integer versioning or

semantic versioning.

This should only be used as a temporary solution - a transition to the versioning scheme used in the CSAF document should be aimed for.

**Number**

The document version of the document this revision history element corresponds to is used as value here.

The value must not be `0` or `0.y.z` if the document status (`/document/tracking/status`) is set to `final` or `interim`.

The value must not include pre-release information and should not include build metadata information.

Further description of this type can be found under types.

**Summary**

The summary aids the reader in initial assessment during inspection.

For the initial version a short notice indicating that this refers to this first version is sufficient,

e.g. “Initial version” or “Initial publication”.

For all subsequent versions the difference to the last published version should be described as concrete as possible.

For pre-release versions, the changes are registered in the entry of the next version to be released.

*Example:*

For version 1.1.0-1 (draft for 1.1.0) the changes are registered in the entry for version 1.1.0.

# 2 Product Tree

The product tree holds all definitions of products referenced later on in the CSAF document.

This is independent of the products’ status.

It is recommended to model a hierarchical structure by means of branches in the `product_tree`.

The branch structure typically consists of vendor, product name and product version.

A product family can also be listed between the vendor and product name. This simplifies grouping and display in a human-readable advisory.

The structure depends strongly on the structure of the products and how they are known by the consumers of the advisories.

*Examples:*

- There is a product family with only 2 products that are relevant for the advisory, then the product family does not necessarily have to be listed.
- For a product family with 120 products for which the advisory applies, the strong recommendation is to list the family as well.

In principle software and hardware components are to be represented separately, since hardware remains the same, even if software is updated.

This also allows separation of software that may run on multiple hardware installations.

The connections of hardware and software are to be represented over `/product_tree/relationships`.

The `product_identification_helper` facilitates the matching between the advisory and the asset or SBOM database.

*Example:*

- If the name of a company changes due to an acquisition, the name would in principle be changed from a certain version of the product.

  In fact, however, other more far-reaching name changes may occur, which should be modelled here.

*Additional for profile CSAF Informational Advisory*

If a `product_tree` is listed in an advisory with profile `CSAF Informational Advisory`, then the listed products are all products to which the advisory applies.

## 2.1 Branches

Branches are used to model a hierarchical product tree.

Further description of this type can be found under types.

## 2.2 Full Product Name

Products which are used in relationships, such as hotfix or operating systems.

Used when branches are not assignable.

Further description of this type can be found under types.

## 2.3 Product Groups

There is no usage documentation yet.

### 2.3.1 Product Groups

Es gibt noch keine Nutzungsdokumentation.

#### *2.3.1.1 Group ID*

Uniquely identifies a group of products throughout the document.

To make it easier to find errors, it is recommended to use a prefix to distinguish from product IDs.

Further description of this type can be found under types.

#### *2.3.1.2 Product ID*

All IDs of products that belong to this group.

Further description of this type can be found under types.

#### *2.3.1.3 Summary*

Describes the common features of the products grouped together in the product group.

Typical summaries are e.g. `End of Life products`, `product families`, `affected products` or `patched products`.

Reduces the effort needed to list products for `remediations`, `flags` and `threats`.

## 2.4 Relationships

Used to model relationships between two products of the `product_tree`.

This could be for example a connection of hard- and software or firmware.

Hardware and corresponding software shall be listed separately with a relationship to connect them.

As a model number (usually) does not change as a result of an update it cannot be determined automatically whether an affected or unaffected part is present.

Thus, a relationship is required to model corresponding software.

A relationship can also be used in cases of OS dependent vulnerabilities or to represent hotfixes.

### 2.4.1 Relationship

A relationship read as `<product_reference> <category> <relates_to_product_reference>`

Relationships are especially useful for hotfixes where the vulnerable product is `installed_with` the fix.

*Example:*

- `product A installed with product B`

#### *2.4.1.1 Category*

Specifies the connection between the named products.

#### *2.4.1.2 Product Reference*

The ID of the product that is stated before the relationship's category.

There must be a corresponding `full_product_name` element with matching `product_id` in the product tree.

Further description of this type can be found under types.

#### *2.4.1.3 Relates To Product Reference*

The ID of the product that is stated after the relationship's category.

There must be a corresponding `full_product_name` element with matching `product_id` in the product tree.

Further description of this type can be found under types.

#### *2.4.1.4 Full Product Name*

Denotes the new product formed by the relationship.

Multiple levels are allowed, such that the new product can be used again in a relationship.

Further description of this type can be found under types.

# 3 Types

This section contains usage information for elements that are used

multiple times throughout the document. To prevent repetitions they will be

described here once. They will be referenced in the document where needed.

**Lang**

The used value should be a valid language code and should not contain subtags reserved for private use.

Subtypes that are reserved for private use (e.g. `qtx`) and `i-default` should not be used.

**Product Group ID**

The same `group_id` must not be defined more than once in the same document.

## 3.1 Product Groups

Allows grouping of products with similarities.

This is useful to reduce the effort required to list products, e.g. for remediations, flags and threats.

**Product ID**

The product ID is used to reference the product later in the document and there is no specified format.

The value is usually assigned by the generator.

To make it easier to find errors, it is recommended to use a prefix to distinguish from product group IDs.

The same `product_id` must not be defined more than once in the same document.

The `product_id` should be referenced somewhere within the same documents.

*Additional for profile CSAF Informational Advisory*

The requirement for the `product_id` to be referenced somewhere within the same documents does not hold for the `csaf_informational_advisory` profile.

Per profile definition the information mentioned in such a CSAF document is applicable to all products listed in the product tree.

## 3.2 Products

There is no usage documentation yet.

**Version usage**

The version must follow either

integer versioning or

semantic versioning

homogeneously within the same document.

## 3.3 Acknowledgments

There is no usage documentation yet.

### 3.3.1 Acknowledgment

An acknowledgment of a contributor or organization that contributed to the document or a vulnerability.

It is important that listed parties agreed to their mention. Mentions based on publicly available papers or the like do not require explicit consent.

In accordance with applicable policies, internal employees of an organization may be listed individually (see example 3.1.1.5 of the specification).

**Organization**

Single organizations or affiliations of names to this organization.

**Summary**

Summarizes the contribution.

#### *3.3.1.1 Names*

Names of the individual contributors, no organization names.

**Name**

The name of the acknowledged person or entity.

#### *3.3.1.2 URLs*

A list of URLs.

**URL**

Link to a contributor's website.

*Example:*

- https://cisa.gov

## 3.4 Branches

Branches are used to model a hierarchical product tree.

### 3.4.1 Branch

The branches are used to model a hierarchical product tree.

Branches are preferred over `full_product_names`.

Products should be listed according to their version.

Specifying exact product versions is preferred over specifying version ranges.

Enumerating the products is usually easier to match on the recipients side.

Additionally, not all version ranges are deterministic.

Since not all required data may yet be available, version ranges are still supported.

If version ranges are used, the Version Range Specifier (VERS) should be preferred.

If this is not possible, the rules of the Version Range Like Specifier (VLS) apply.

It is possible that the exact version that is affected is not known. In this case a version range can be used.

*Example:*

Version 4.2 of a product may be affected and version 4.3 contains a patch.

- Version 4.3 listed as fixed product version
  - if it is not known since which version the product is affected
    - <=4.2 (range) listed as affected
    - or all versions below listed individually
  - if it is known since which version the product is affected, the given range or listed versions should be adjusted accordingly

Hotfixes should be represented through the relationships. The product without and with hotfix are listed, the first in the `affected` product status group, the latter in the `fixed` group.

**Category**

Specifies how the value in the adjacent name field should be understood.

If the value is `product_version`, the property `name` must not contain a version range. The following patterns are considered to indicate a version range:

```
* `<`

* `<=`

* `>`

* `>=`

* `after`

* `all`

* `before`
```

* `earlier`

* `later`

* `prior`

* `versions`

**Name**

The value is further defined by the branch's category.

If the `category` is `product_version_range`, the `name` should conform to the VERS specification, i.e. match the regex `^vers:[a-z\\.\\-\\+][a-z0-9\\.\\-\\+]*/.+`

#### *3.4.1.1 Product*

A leaf in the hierarchical structure of branches, includes exactly one product.

This product can be a single version or a range of versions.

Further description of this type can be found under types.

## 3.5 Full Product Name

The property `product_identification_helper` should exist and contain at least one item.

**Name**

The name of the product as used in a human-readable advisory.

In the branch hierarchy this name can be formed by stringing the `name` elements along the path to the leaf.

**Product Id**

Further description of this type can be found under types.

### 3.5.1 Product Identification Helper

A collection of different possibilities to identify a product in the asset database or SBOM.

When it is used to identify a new product formed by a relationship no value must be the same to any of the products that make up the relationship.

Otherwise, a matching would return misleading results.

*Example*:

There is an application (`Product A`) that has the same hash on Linux and Windows and a vulnerability is found that affects the program when being executed on Windows but not on Linux.

If the same hash would be linked in the relationship that forms the products `Product A installed on Windows` and `Product A installed on Linux`, a matching algorithm would return the same list of assets for all 3 product IDs.

However, that is an ambiguous match.

**Cpe**

Common Platform Enumeration, see cpe.mitre.org.

Can be CPE 2.2 and CPE 2.3, while 2.3 is preferred.

When using CPE, please be aware of the limitations and that it might be deprecated in the future.

**Purl**

A purl is a URL composed of seven components which are separated by a specific character for unambiguous parsing:

```
scheme:type/namespace/name@version?qualifiers#subpath
```

The definition for each component is:

- `scheme`: this is the URL scheme with the constant value of "pkg". One of the primary reason for this single scheme is to facilitate the future official registration of the "pkg" scheme for package URLs. Required.
- `type`: the package "type" or package "protocol" such as maven, npm, nuget, gem, pypi, etc. Required.
- `namespace`: some name prefix such as a Maven groupid, a Docker image owner, a GitHub user or organization. Optional and type-specific.
- `name`: the name of the package. Required.
- `version`: the version of the package. Optional.
- `qualifiers`: extra qualifying data for a package such as an OS, architecture, a distro, etc. Optional and type-specific.
- `subpath`: extra subpath within a package, relative to the package root. Optional.

Components are designed such that they form a hierarchy from the most significant component on the left to the least significant component on the right.

A purl must NOT contain a URL Authority i.e. there is no support for username, password, host and port components.

A namespace segment may sometimes look like a host but its interpretation is specific to a type.

Also, a purl does not contain double slashes `//`, that is the scheme is `pkg:`.

There

Purls are particularly well suited for identifying components within the package types supported by purl.

#### *3.5.1.1 Hashes*

Hashes are particularly well suited for the identification of software.

##### 3.5.1.1.1 Hash

There is no usage documentation yet.

**Filename**

Specifies the file name over which the hash was formed.

###### 3.5.1.1.1.1 File Hashes

Since collision attacks exist for `md5` and `sha1`, these should not be the only hash algorithms present.

Either of those should be accompanied by a second cryptographically stronger hash.

The same hash algorithm must not be used in multiple items in one item of hashes belonging to one file.

###### 3.5.1.1.1.1.1 File Hash

The hash of a file consisting of the used algorithm and resulting hash value.

**Algorithm**

Cryptographically secure algorithms should be preferred to avoid misidentification through hash collision.

Currently, the algorithms SHA256, SHA512 and SHA3 are recommended.

**Value**

The hash value should not be shorter than 64 characters.

#### *3.5.1.2 Model Numbers*

Model numbers are particularly well suited for the identification of hardware.

Software installed on the hardware must be mapped via relationships (`/product_tree/relationships`).

**Model Number**

Should not be version strings of products.

#### *3.5.1.3 SBOM Urls*

There is no usage documentation yet.

**SBOM Url**

If a SBOM is offered for download (publicly or protected), the URL can serve as a unique identifier for the version of the product that the SBOM describes.

If it links to a protected area, the URL must still point to the canonical download value (not just to the portal where it is retrieved), otherwise uniqueness is lost.

A SBOM URL should not be used in combination with a product version range.

#### *3.5.1.4 Serial Numbers*

Serial numbers are particularly well suited for the identification of hardware.

Software installed on that hardware must be mapped via relationships (`/product_tree/relationships`).

Serial numbers are especially useful for end-of-life products, or if a series of product instances cannot be provided with an update due to production or technical characteristics.

**Serial Number**

There is no usage documentation yet.

#### *3.5.1.5 SKUs*

SKUs are particularly well suited for products in direct sales.

SKUs provide the customer with the option to identify the product via the information available from the ordering process.

**SKU**

There is no usage documentation yet.

#### *3.5.1.6 Generic URIs*

Generic URIs are used to identify components within a SBOM (especially relevant for the VEX use case).

For CycloneDX and SPDX examples refer to the specification.

It can also be used for manufacturer-specific product identifiers not yet supported.

#### 3.5.1.6.1 Generic URI

There is no usage documentation yet.

**Namespace**

Specifies the URL where the value of the URI is valid and describes how to use the value.

See examples 16 and 17 from the specification.

**URI**

Identifier for a product that is valid in the context of the specification given under `namespace`.

## 3.6 Notes

A note with category `summary` should consist of two to three sentences with barely any technical detail.

A note with category `description` should consist of one to three paragraphs with possibly some technical detail.

A note with category `details` should contain a detailed description, exceeding the length of a `description` and go deeper into technical details.

Basically, at least one summary should be listed, a description is preferred.

### 3.6.1 Note

There is no usage documentation yet.

**Audience**

Describes for whom this text was written.

*Examples:*

- `all`
- `executives`
- `operational management and system administrators`
- `safety engineers`

Should be specified in particular if different target groups are addressed in the CSAF document and different texts exist for these target groups.

*Example:*

An executive summary will have different content than a note on technical details.

**Category**

Defines the general category of the note.

**Text**

Content of the note. Scope and depth are determined by the note's category.

The format can be plain text or markdown. HTML is not allowed.

**Title**

Should describe the content of the note concisely.

## 3.7 References

There is no usage documentation yet.

### 3.7.1 Reference

There is no usage documentation yet.

#### *3.7.1.1 Category*

The category of a reference should always be specified to be explicit.

`external`: links e.g. to a support article describing installation and update procedures, where descriptions of hotfixes can be found or to a description of restart procedures

`self`: links e.g. to exactly the same vulnerability in the human readable advisory

For vulnerabilities, usually the category refers to an `external` reference.

For the document references with category `self` are more common.

In the document references the canonical URL and reference of human-readable advisory must have category `self`.

All other references usually are `external`.

#### *3.7.1.2 Summary*

A meaningful title or short summary of the reference.

For web pages it's often the page title from the HTML metadata.

For other documents it's the document's title.

For different formats of advisories the title or summary has the following format:

`ID: TITLE - FORMAT`

where `ID` is the value of `/document/tracking/id`, `TITLE` is optional and the value of `/document/title` and `FORMAT` is one of `PDF`, `TXT`, `CSAF` or `HTML`.

If different versions of the document reside under different URLs, the value of `/document/tracking/version` should be appended as `- version VERSION` after `FORMAT`.

If the latest version is always at the same URL, the `VERSION` part can be identified as `- latest version`

#### *3.7.1.3 Url*

The URL of the reference.

# 4 Vulnerabilities

The list of vulnerabilities contains all the information that helps to describe vulnerabilities and their impact and remediations.

*Additional for profile CSAF Informational Advisory*

The `csaf_informational_advisory` profile deals with information that are not classified as vulnerabilities.

Therefore, it must not have the `/vulnerabilities` element.

If there is a vulnerability, another profile, e.g. `csaf_security_advisory` should be selected.

*Additional for profile CSAF Security Advisory*

The `csaf_security_advisory` profile deals with vulnerabilities.

Therefore, it must have the `/vulnerabilities` element.

*Additional for profile CSAF VEX*

The `csaf_vex` profile deals with vulnerabilities.

Therefore, it must have the `/vulnerabilities` element.

## 4.1 Vulnerability

It is recommended to provide a CVE number (`cve`) to support the users efforts to find more details about a vulnerability and potentially track it through multiple advisories.

If no CVE exists for that vulnerability, it is recommended to get one assigned.

A CWE number (`cwe`) should be given.

*Additional for profile CSAF Security Advisory*

For the `csaf_security_advisory` profile there must be a `notes` and `product_status` property for a vulnerability.

*Additional for profile CSAF VEX*

For the `csaf_vex` profile there must be a `notes` and `product_status` property for a vulnerability.

For the `csaf_vex` profile at least one of the properties `cve` or `ids` must be present.

**CVE**

CVE numbers help to identify the vulnerability and ensure the same terms are used.

The same CVE must not be repeated for multiple vulnerability items.

CVE numbers should also be assigned for vulnerabilities found internally.

CVE numbers can be requested from MITRE or another CNA (CVE Numbering Authority).

If organizations regularly publish vulnerabilities, they should consider becoming a CNA, as described here.

**Discovery Date**

The discovery date specifies when the vulnerability was discovered.

For logical reasons the value can not be a timestamp in the future.

### 4.1.1 Notes

The notes should contain information about the vulnerability.

This can be available in different levels of detail.

A Note adding a CVE description is recommended. The title of this note should be `CVE description` and the category should be set as `description`.

For a summary of the vulnerability, a note with title `Vulnerability summary` and category `summary` should be created.

The impact of the vulnerability should be listed as a threat (`/vulnerabilities[]/threats`) with category `impact`.

For vulnerabilities there should be no note with category `legal_disclaimer`.

For the `csaf_security_advisory` profile `notes` must exist.

For the `csaf_vex` profile `notes` must exist.

Further description of this type can be found under types.

*Additional for profile CSAF Security Advisory*

For the `csaf_security_advisory` profile `/vulnerabilities[]/notes` must exist.

*Additional for profile CSAF VEX*

For the `csaf_vex` profile `/vulnerabilities[]/notes` must exist.

### 4.1.2 References

A reference to the JSON of the CVE entry should be listed.

Additional information about the vulnerability can be listed:

- additional help articles on the vulnerability
- Knowledge Base articles
- link to the bug tracking issue
- link to a pull/merge request if publicly available
- ...

A website of the discoverer of the vulnerability may also be linked.

Further description of this type can be found under types.

**Release Date**

Specifies since when the vulnerability has been publicly known.

**Title**

The title of the vulnerability.

### 4.1.3 Acknowledgments

Acknowledgements of contributors specific to this vulnerability.

If a party would be mentioned in all vulnerabilities of the advisory, it should be considered to mention it in `/document/acknowledgements`.

A double mention is valid, though.

The same notes apply as for `/document/acknowledgments`.

Further description of this type can be found under types.

### 4.1.4 CWE

CWE is a community-developed list of software and hardware weakness types.

It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

(Cited from cwe.mitre.org)

The Vulnerability classification of CWE helps assess groups of vulnerabilities.

The CWE that most accurately identifies the vulnerability should be chosen and the given CWE must exist and be valid.

A link to the used CWE should be provided in the vulnerability's references.

**Id**

The ID from the CWE catalogue.

**Name**

The full name as given in the specification.

*Examples:*

- `CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')`

### 4.1.5 Flags

Machine-readable assessment of why a product is not affected.

Should not be set without a human-readable impact statement (in `/vulneratbilities[]/threats`).

The use of flags is recommended for more mature ProductCERTs.

May be extended in later versions of the standard.

#### *4.1.5.1 Flag*

Each flag must include at least one of the elements `group_ids` or `product_ids`.

Using the `product_ids` is slightly preferred.

A product must not be listed as member of more than one flag item with a VEX justification code.

This also includes indirect relations through product groups.

**Date**

The date when the flag was set or an analysis was carried out that led to setting the flag.

**Group IDs**

List of group IDs to which the flag applies.

Further description of this type can be found under types.

**Label**

Describes the most appropriate reason why the vulnerability is not exploitable in machine-readable form.

If there is no machine-readable status justification available that fits the reason, the impact statement must be done through an element in `/vulnerabilities[]/threats` with category `impact` and the description in `details`.

**Product Ids**

List of product IDs to which the flag applies.

Further description of this type can be found under types.

### 4.1.6 Ids

Lists more IDs apart from CVE.

*Examples:*

- ID of a ticket in a bug tracker
- ID of a Pull Request in a code repository

#### *4.1.6.1 ID*

**System Name**

Specifies the environment in which the ID under `text` is valid.

**Text**

Ticket number or identifier in the bug tracker.

It can also be an ID of support tickets or knowledge base articles.

The value should not be a CVE ID.

The CVE ID for a vulnerability must be listed in the `cve` property.

### 4.1.7 Involvements

The list of involvements must not contain the same `party` regardless of its `status` more than once at any `date`.

Can be used to document the disclosure timeline.

Who contacted whom and when, who learned about the vulnerability and what actions were taken.

Should be used especially if the vendor did not respond to the contact attempts or denies the existence of the vulnerability or the CVD process failed.

#### *4.1.7.1 Involvement*

For temporal classification, a `date` should be given.

**Date**

The date when the action happened, the date when the contact was made, or the date when the producer contacted back and rejected the report.

**Party**

Defines the category to which the party belongs about which a statement is made.

**Status**

Defines the contact status the party is in at the time of writing.

- `in_progress`:

  Is suitable in many situations, e.g. when there is a hotfix, when feedback has been received, when more information is needed, …

- `contact_atempted`:

  This status indicates that a contact has been attempted. If possible, it should be documented how the contact was attempted.

- `not_contacted`:

  It should be documented why no contact was made.

- `disputed`:

  It should be indicated why the party does not consider this to be a vulnerability, or why the vulnerability has no implication on security.

- `completed`:

  The party asserts investigation of the vulnerability to be complete.

- `open`

  The party has acknowledged awareness of the vulnerability report.

The statuses `contact_atempted` and `not_contacted` should not be specified by the vendor.

**Summary**

A summary that briefly describes what the status is all about.

### 4.1.8 Product Status

The product status provides information on the status products in the `product_tree` are in.

Generally, vulnerable (`known_affected`) and fixed (`fixes`) versions of products should be given.

The different lists of product status groups contain the IDs of corresponding products.

The same product ID must not be member of contradicting product status groups.

I.e. a product listed in the group of *affected* products must not be listed again in the group of *not affected* products.

The contradicting groups are

- *affected*
  - `product_status/first_affected[]`
  - `product_status/known_affected[]`
  - `product_status/last_affected[]`
- *not affected*
  - `product_status/known_not_affected[]`
- *fixed*
  - `product_status/first_fixed[]`
  - `product_status/fixed[]`
- *under investigation*
  - `product_status/under_investigation[]`

Each product listed in the *affected* group should be referenced in at least one remediation item (`/vulnerabilities[]/remediations`).

For each product listed in the *not affected* group, the reason why it is not affected should be given.

This reason, also called impact statement, should be listed in the threats (`/vulnerabilities[]/threats`) with category `impact` or as machine-readable status justification in the flags (`/vulnerabilities[]/flags`).

It is recommended to use both, when applicable.

The status `known_not_affected` is used in the context of security advisories to exclude individual products. This is mainly used for products one could assume to be affected.

To define general non-affectedness, the VEX profile should be used.

An issuer might recommend (`product_status/recommended`) a product version from any group.

If more than one product version is patched, the author should also specify which version is recommended.

A recommended product version can also come from the *affected* group, i.e. if it was discovered that fixed versions introduce a more severe vulnerability.

If there are two patched versions of a vulnerability that contain different vulnerabilities, the `recommended` field can also be used to specify which version is recommended for use.

Version boundaries can be represented in the `first_*` and `last_*` fields.

For simplification, elements from `first_*` and `last_*` should also be listed in the main groups `known_*`.

For products listed in `under_investigation` it is expected that the outcome of analysis are reported in a later version of the document.

For the `csaf_security_advisory` profile the property `product_status` must exist.

For the `csaf_vex` profile at least one of the elements `fixed`, `known_affected`, `known_not_affected` or `under_investigation` must be present.

**First Affected**

The product version that was affected first or with which product version the vulnerability was introduced.

All previous versions are assumed to be unaffected.

The first affected version can also be an “End of Life” version.

A product listed here must not be listed in any of the following

- `product_status/known_not_affected[]`
- `product_status/first_fixed[]`
- `product_status/fixed[]`

- `product_status/under_investigation[]`

For each product ID listed here, a corresponding remediation should exist.

For each product ID listed here, a score object which covers the product should exist.

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.

**First Fixed**

The version of a product which first contains a fix, or which product version closed the vulnerability.

All previous versions are assumed to be affected.

A product listed here must not be listed in any of the following

- `product_status/first_affected[]`
- `product_status/known_affected[]`
- `product_status/last_affected[]`
- `product_status/known_not_affected[]`
- `product_status/under_investigation[]`

For each product ID listed here, a CVSS applying to this product should have an environmental score of `0`.

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.

**Fixed**

Listing fixed products facilitates the examination if products are affected.

A product listed here must not be listed in any of the following

- `product_status/first_affected[]`
- `product_status/known_affected[]`
- `product_status/last_affected[]`
- `product_status/known_not_affected[]`
- `product_status/under_investigation[]`

For each product ID listed here, a CVSS applying to this product should have an environmental score of `0`

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.

**Known Affected**

Listing affected products facilitates the examination if products are affected.

A product listed here must not be listed in any of the following

- `product_status/known_not_affected[]`
- `product_status/first_fixed[]`
- `product_status/fixed[]`
- `product_status/under_investigation[]`

For each product ID listed here, a remediation should exist.

For each product ID listed here, a score object which covers the product should exist.

For all profiles other than `csaf_vex`, an action statement should be listed in the remediations - also for products for which there is no (planned) fix.

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

For the `csaf_vex` profile, for each item in `known_affected` a corresponding action statement in `/vulnerabilities[]/remediations` must exist.

Further description of this type can be found under types.

*Additional for profile CSAF VEX*

For the `csaf_vex` profile, for each item in `known_affected` a corresponding action statement in `/vulnerabilities[]/remediations` must exist.

The action statement describes what a user should do.

Optional, additional information MAY also be provided through `/vulnerabilities[]/notes` and `/vulnerabilities[]/threats`.

**Known Not Affected**

Lists only non-affected products.

Products that are only not affected in a certain configuration must not be listed here.

These products are considered affected as the user has to check the configuration (which is a remediation and can be described as action statement).

It should always be stated why the products are not affected.

A product listed here must not be listed in any of the following

- `product_status/first_affected[]`
- `product_status/known_affected[]`
- `product_status/last_affected[]`
- `product_status/first_fixed[]`
- `product_status/fixed[]`
- `product_status/under_investigation[]`

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

For the `csaf_vex` profile, for each item in `known_not_affected` a corresponding impact statement must exist in `/vulnerabilities[]/flags` or `vulnerabilities[]/threats`.

For the latter one, the `category` value for such a statement must be `impact`

Further description of this type can be found under types.

*Additional for profile CSAF VEX*

For the `csaf_vex` profile, for each item in `known_not_affected` a corresponding impact statement must exist in `/vulnerabilities[]/flags` or `vulnerabilities[]/threats`.

For the latter one, the `category` value for such a statement must be `impact` and the `details` field SHALL contain a description why the vulnerability cannot be exploited.

**Last Affected**

Describes the last version that is affected by the vulnerability.

All subsequent versions are either fixed or not affected.

A product listed here must not be listed in any of the following

- `product_status/known_not_affected[]`
- `product_status/first_fixed[]`
- `product_status/fixed[]`
- `product_status/under_investigation[]`

For each product ID listed here, a remediation should exist.

For each product ID listed here, a score object which covers the product should exist.

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.

**Recommended**

If there are multiple fixes available, it is recommended to list the preferred one here.

If there is no fixed version, but several vulnerable ones where the characteristics of the vulnerability differs, the field can also be used to recommend less vulnerable versions.

If a patch fixes one vulnerability but introduces another, it is necessary to weigh which of the vulnerabilities is more acceptable and recommend the less vulnerable version accordingly.

Whenever a vulnerable version is specified as recommended, a textual description under `/vulnerabilities[]/notes` must indicate why it was recommended.

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.

**Under Investigation**

For the products listed here, it is not yet clear whether they are affected or not affected.

It is expected that in a later release the analysis result will be communicated.

If general mitigation guidance can be provided and is applicable to products currently under investigation, it should be provided.

A valid remediation can also be of the category `none_available` or `no_fix_planned`.

Especially useful in cases of large Multi-Party Coordinated Vulnerability Disclosure to provide the current status of the investigation.

A product listed here must not be listed in any of the following

- `product_status/first_affected[]`
- `product_status/known_affected[]`
- `product_status/last_affected[]`
- `product_status/known_not_affected[]`
- `product_status/first_fixed[]`

- `product_status/fixed[]`

For each entry a corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.

### 4.1.9 Remediations

All information on how to fix the vulnerability, or even if no fix is possible.

For End of Life products, typically, the category `no_fix_planned` is used.

If no patch or other remediation is available yet, this should be communicated via `none_available`.

It should then be described why this is the case.

#### *4.1.9.1 Remediation*

Each remediation must include at least one of the elements `group_ids` or `product_ids`.

**Category**

The category indicates to which type the remediation belongs.

For end of life products the category `no_fix_planned` is common. A reference to the fact that these are end of life products is sufficient.

It is best to include the date since when these products are no longer supported. A reference to compatible successor products is permissible and recommended.

In other cases where `no_fix_planned` is used, it should be stated why no fix has been created or is planned.

Permissible reasons include that the hardware requirements of the mitigation measure exceed the capabilities of the existing hardware.

*Example:*

- An embedded system uses 1024 RSA keys when RSA 3072 keys would be secure but there is not enough memory to execute the computations.

The value `none_available` must always be specified if no remediation is available.

Can also be specified if no patch is available yet, but it is planned. In this case an expected release date of the patch should be specified.

Once the patch is available, a remediation with the category `vendor_fix` will replace this remediation.

Unless otherwise specified, the category `vendor_fix` assumes that the patch fully remediates the vulnerability.

If there are any effects on functionality, these must be entered in the details of the remediation.

Additionally, installation instructions or prerequisites can also be specified.

For the value `mitigation` it is crucial that the exploitability is reduced, but the vulnerability is not completely fixed.

A `workaround` describes a configuration or special deployment that can be used to prevent the exploitability of the vulnerability.

Hotfixes are usually classified as `workarounds` because they prevent exploitability.

However, hotfixes can also be classified as `mitigation` if they do not completely fix the vulnerability.

Hotfixes can also be listed at `vendor_fix` if they completely fix the problem.

**Date**

The date from when on the remediation is/was available.

This information is helpful in audits to prove from when on one was aware of a certain remediation measure.

**Details**

The detailed description of the remediation.

The content is determined by the category.

#### 4.1.9.1.1 Group Ids

List of group IDs to which the remediation applies.

Further description of this type can be found under types.

**Product Ids**

List of product IDs to which the remediation applies.

Further description of this type can be found under types.

**Url**

The URL where the remediation can be retrieved.

#### 4.1.9.1.2 Entitlements

Conditions that must be fulfilled in order to receive the patch or implement the mitigation action.

*Example:*

An active support contract is required.

**Entitlement**

There is no usage documentation yet.

#### 4.1.9.1.3 Restart Required

Specifies whether a restart is required for the mitigation measure, and if so, in what scope.

This is relevant as in some cases where connections exist the scope may be beyond a single machine.

This information may help in making better / risk-based decisions.

A remediation with `restart_required/category` `none` may be applied faster than another one.

The information is also especially relevant if the product itself is modified with the mitigating measure (workaround, mitigation, or vendor fix).

If the product environment is modified (e.g., adjusting the rules of an external firewall), it is difficult if not impossible for vendors to determine if a restart is required.

**Category**

Specifies the type and scope of the required restart in machine-readable form.

Mitigation measures that do not require a restart and can therefore be executed during operation may be preferred.

- `none`: No restart is necessary.

  Example: “kill telnet process”.

- `vulnerable_component`: Only the affected product needs to be restarted.

  Example: natepad++ portable, exit, install new version or patch, and restart.

- `service`: Services need to be restarted.

  Example: `CVE 2019-11043` remote code execution in a configuration of nginx and fpn. Service must be restarted when running nextcloud in default configuration.

- `parent`: Is always the case when the vulnerability is in a product running in a child process. If only the child program is restarted, the parent program does not know that a new binary exists.

- `dependencies`: The vulnerability in the program in the parent process and all programs depending on it must be restarted.

  Example: pppd

- `connected`: All products connected to the affected product, as well as the affected product itself, must be restarted. The connection can be via network or other inter-process communication.

  Example: A product that uses TLS only at the beginning and does not do session renegotiation.

- `machine`: The entire machine must be rebooted.

  Example: Windows 10 Semiannual Upgrade.

- `zone`: A zone consisting of multiple machines (e.g. security zone or broadcast zone) must be restarted.

  Example: WannaCry infecting industrial PCs which have the Write Filter enabled. Windows reboots after infection when WannaCry tries to encrypt the disk which removes WannaCry from this PC.

  However, other PCs in the same security zone still have WannaCry and reinfect each other. To get a clean zone, all equipment must be turned off and be rebooted afterwards.

- `system`: A system usually consisting of multiple zones must be restarted.

  Example: Update to a new (encrypted) version of an industry protocol.

**Details**

The details should describe which effects a restart has, or which procedures are to be considered with a restart.

Further descriptions of the scope of the restart should also be added here.

### 4.1.10 Scores

Provides an assessment of the severity of the vulnerabilities.

The same product must not be listed more than once with a CVSS-Vector with the same version.

Fixed products should not be listed here.

#### *4.1.10.1 Score*

For CVSS scores, the referenced schema must be adhered to.

For CVSS scores, the computed values must be correct according to their definition. For this computation, the `vectorString` takes precedence over the other properties.

For CVSS scores, the given properties must not contradict the CVSS vector.

CVSS v2 (`cvss_v2`) should not be the only scoring item used.

Instead of using CVSS v3.0, an upgrade to CVSS v3.1 should be considered.

Generally, it is recommended to use CVSS v3.1.

**Products**

Specifies the products to which the score applies.

If the score applies to all affected products within the vulnerability, a score element whose products resemble the *affected* product group should be specified.

Further description of this type can be found under types.

**Cvss V2**

The schema at https://www.first.org/cvss/cvss-v2.0.json must be adhered to.

The computed values must be correct according to their definition. For this computation, the `vectorString` takes precedence over the other properties.

The given properties must not contradict the CVSS vector.

If the corresponding product is listed in `/vulnerabilities[]/product_status/first_fixed[]` or `/vulnerabilities[]/product_status/fixed[]`, the `environmental_score` should be `0`.

The `exploitability` should be given to convey at least if the vulnerability is already actively exploited.

The single fields are encoded in the vector string and thus may be redundant.

However, adding them is encouraged as the vector string may be incomplete.

Using CVSS v3.1 is encouraged.

**Cvss V3**

The schema at https://www.first.org/cvss/cvss-v3.1.json respectively https://www.first.org/cvss/cvss-v3.0.json must be adhered to.

The computed values must be correct according to their definition.

For this computation, the `vectorString` takes precedence over the other properties.

The given properties must not contradict the CVSS vector.

If the corresponding product is listed in `/vulnerabilities[]/product_status/first_fixed[]` or `/vulnerabilities[]/product_status/fixed[]`, the `environmental_score` should be `0`.

The `exploitCodeMaturityType` should be given to convey at least if the vulnerability is already actively exploited.

The single fields are encoded in the vector string and thus may be redundant.

However, adding them is encouraged as the vector string may be incomplete.

Using CVSS v3.1 is encouraged.

### 4.1.11 Threats

Lists existing or possible threats like known exploits.

If a product is not affected an impact statement (threat with category `impact`) should be listed explaining the reason.

The threats should contain information about a vulnerability that is particularly exploited in a specific geographic area or industry (e.g. banks).

As the listed threats contains information that may change over time it should be assumed that the information is/was up-to-date at the time of writing.

#### *4.1.11.1 Threat*

There is no usage documentation yet.

**Category**

Specifies the type of information.

**Date**

The date when the information was entered, or was current.

**Details**

A detailed description of the information determined by the category.

**Group Ids**

The product groups which the information refers to.

Further description of this type can be found under types.

**Product Ids**

The products to which the information refers to.

A corresponding `full_product_name` element with matching `product_id` must be defined.

Further description of this type can be found under types.