# CSAF Common Vulnerability Reporting Framework (CVRF) Version 1.2

## Working Draft 01

## 10 March 2017

**Technical Committee:**

OASIS Common Security Advisory Framework (CSAF) TC

**Chair:**

Omar Santos (os@cisco.com), Cisco Systems

**Editor:**

Stefan Hagen (stefan@hagen.link), Individual

**Additional artifacts:**

This prose specification is one component of a Work Product that also includes:

- XML schemas: http://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csd01/schemas/

**Related work:**

This specification replaces or supersedes:

- *The Common Vulnerability Reporting Framework (CVRF) Version 1.1*. May 2012. ICASI. http://www.icasi.org/the-common-vulnerability-reporting-framework-cvrf-v1-1/.

**Declared XML namespaces:**

- http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/common
- http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf
- http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/prod
- http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/vuln

**Abstract:**

The CSAF Common Vulnerability Reporting Framework (CVRF) Version 1.2 is the definitive reference for the CVRF language.

**Status:**

**URI patterns:**

Initial publication URI:
http://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csd01/csaf-cvrf-v1.2-csd01.docx
Permanent “Latest version” URI:
http://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csaf-cvrf-v1.2.docx

(Managed by OASIS TC Administration; please don’t modify.)

# Table of Contents

# 1 Introduction

The CSAF Common Vulnerability Reporting Framework (CVRF) is a language to exchange Security Advisories formulated in XML.

This document is the definitive reference for the CVRF Dictionary of Elements version 1.2. The encompassing XML schema files noted in the Additional Artifacts section of the title page shall be taken as normative in the case a gap or an inconsistency in this explanatory document becomes evident.

The version 1.2 of the CVRF allows the users to transition from Common Vulnerability Scoring System (CVSS) version 2 to version 3 as it supports both CVSS versions.

Please note that CVRF 1.2 is **not** backward compatible with CVRF 1.1 published by the Internet Consortium for Advancement of Security on the Internet (ICASI) and contributed to OASIS for future evolution by the Common Security Advisory Framework (CSAF) TC.

The following presentation is grouped by schema, and is not simply derivative documentation from the schema documents themselves. The information contained aims to be more descriptive and complete. Where applicable, common conventions are stated and known common issues in usage are pointed out informatively to support implementers of document producers and consumers alike.

From a high-level perspective, any Security Advisory purported by a CVRF version 1.2 adhering document (inside the root `cvrf:cvrfdoc` element) must provide at least the following top-level elements in the displayed sequence (Format is "**Concept**: `namespace:Element`"):

| | | |
|---|---|---|
| 1. | **Title**: | `cvrf:DocumentTitle` |
| 2. | **Type**: | `cvrf:DocumentType` |
| 3. | **Publisher**: | `cvrf:DocumentPublisher` |
| 4. | **Tracking**: | `cvrf:DocumentTracking` |

This minimal required set does not provide any useful info on products, vulnerabilities, or security advisories, thus a maximal top-level set of elements is given here below:

| | | |
|---|---|---|
| 1. | **Title**: | `cvrf:DocumentTitle` |
| 2. | **Type**: | `cvrf:DocumentType` |
| 3. | **Publisher**: | `cvrf:DocumentPublisher` |
| 4. | **Tracking**: | `cvrf:DocumentTracking` |
| 5. | **Notes**: | `cvrf:DocumentNotes` |
| 6. | **Distribution**: | `cvrf:DocumentDistribution` |
| 7. | **Aggregate Severity**: | `cvrf:AggregateSeverity` |
| 8. | **References**: | `cvrf:DocumentReferences` |
| 9. | **Acknowledgements**: | `cvrf:Acknowledgements` |
| 10. | **Product Tree**: | `prod:ProductTree` |
| 11. | **Vulnerability**: | `vuln:Vulnerability` |

Care has been taken, to design the containers for product and vulnerability info to support fine-grained mapping of security advisories onto product and vulnerability and minimize data duplication through referencing.

The display of the elements representing **Product Tree** and **Vulnerability** info has been placed in the sections named accordingly.

General design considerations that place CSAF CVRF version 1.2 in the wider context of security advisories are to be found in the section Design Considerations.

As the XML format is not primarily targeting human readers but more programs parsing, validating and transforming no example is given in this introduction but instead examples derived from several real-world security advisories are stated in the non-normative appendix.

## 1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 1.2 Normative References

| | |
|---|---|
| **[RFC2119]** | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt. |
| **[XML]** | Extensible Markup Language (XML) 1.0 (Fifth Edition), T. Bray, J. Paoli, M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, W3C Recommendation, November 26, 2008, http://www.w3.org/TR/2008/REC-xml-20081126/. Latest version available at http://www.w3.org/TR/xml. |
| **[XML-Schema-1]** | W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures, S. Gao, M. Sperberg-McQueen, H. Thompson, N. Mendelsohn, D. Beech, M. Maloney, Editors, W3C Recommendation, April 5, 2012, http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/. Latest version available at http://www.w3.org/TR/xmlschema11-1/. |
| **[XML-Schema-2]** | W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, D. Peterson, S. Gao, A. Malhotra, M. Sperberg-McQueen, H. Thompson, Paul V. Biron, Editors, W3C Recommendation, April 5, 2012, http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/. Latest version available at http://www.w3.org/TR/xmlschema11-2/. |

## 1.3 Non-Normative References

| | |
|---|---|
| **[CPE23-N]** | Common Platform Enumeration: Naming Specification Version 2.3, B. Cheikes, D. Waltermire, K. Scarfone, Editors, NIST Interagency Report 7695, August 2011, http://dx.doi.org/10.6028/NIST.IR.7695. |
| **[CPE23-M]** | Common Platform Enumeration: Naming Matching Specification Version 2.3, M. Parmelee, H. Booth, D. Waltermire, K. Scarfone, Editors, NIST Interagency Report 7696, August 2011,http://dx.doi.org/10.6028/NIST.IR.7696. |
| **[CPE23-D]** | Common Platform Enumeration: Dictionary Specification Version 2.3, P. Cichonski, D. Waltermire, K. Scarfone, Editors, NIST Interagency Report 7697, August 2011, http://dx.doi.org/10.6028/NIST.IR.7697. |
| **[CPE23-A]** | Common Platform Enumeration: Applicability Language Specification Version 2.3 (NISTIR 7698), D. Waltermire, P. Cichonski, K. Scarfone, Editors, NIST Interagency Report 7698, August 2011, http://dx.doi.org/10.6028/NIST.IR.7698. |
| **[CVSS2]** | A Complete Guide to the Common Vulnerability Scoring System Version 2.0, P. Mell, K. Scarfone, S. Romanosky, Editors, First.org, Inc. June 2007, https://www.first.org/cvss/cvss-v2-guide.pdf. |
| **[CVSS3]** | Common Vulnerability Scoring System v3.0: Specification Document, FIRST.Org, Inc., December 2015, https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf. |
| **[DCMI11]** | DCMI Metadata Terms v1.1, Dublin Core Metadata Initiative, DCMI Rec., June 14, 2012, http://dublincore.org/documents/2012/06/14/dcmi-terms/. Latest version available at http://dublincore.org/documents/dcmi-terms/. |
| **[SCAP12]** | The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2, D. Waltermire, S. Quinn, K. Scarfone, A. Halbardier, Editors, NIST Spec. Publ. 800-126 rev. 2, September 2011, http://dx.doi.org/10.6028/NIST.SP.800-126r2. |

## 1.4 Typographical Conventions

Keywords defined by this specification use this `monospaced` font.

```
Normative source code uses this paragraph style.
```

Some sections of this specification are illustrated with non-normative examples.

*Example 1: text describing an example uses this paragraph style*

```
Non-normative examples use this paragraph style.
```

All examples in this document are non-normative and informative only.

Representation-specific text is indented and marked with vertical lines.

> ### Representation-Specific Headline
> Normative representation-specific text

All other text is normative unless otherwise labeled.

# 2 Design Considerations

A Security Advisory defined as a CSAF CVRF document is the result of complex orchestration of many players and distinct and partially difficult to play schemas.

Historically the format was chosen as [XML] based on a small set of XSD ([XML-Schema-1], [XML-Schema-2]) schema files spanning a combined namespace anchored at a single root. This was even more so natural, as it aligned well with separation of concerns and shared the format family of information interchange used in the supplying industry (product and vulnerability information).

The acronym CSAF: "Common Security Advisory Framework" stands for the target of concerted mitigation and remediation accomplishment and the name part CVRF: "Common Vulnerability Reporting Framework" reflects the origin of the orchestrating tasks.

Technically the use of XML allows validation and proof of model conformance (through established schema based validation) of the declared information inside CVRF documents.

## 2.1 Construction Principles

The CSAF CVRF schema structures its derived documents into three main classes of the information conveyed:

1. The frame, aggregation, and reference info of the document
2. Product information considered relevant by the creator
3. Vulnerability information and its relation to the products declared in 2.

The prescribed sequence of ordered elements inside these main classes (containers) has been kept stable (e.g. no lexical sorting of sequences) to reduce the amount of changes required for upgrading.

Wherever possible repetition of data has been replaced by linkage through ID elements. Consistency on the content level thus is in the responsibility of the producer of such documents, to link e.g. vulnerability info to the matching product.

A dictionary like presentation of all defined elements of the schemas is given in the following sections. Any expected relations to other elements (linkage) is described there. This linking relies on setting attribute values accordingly (mostly guided by industry best practice and conventions) and thus implies, that any deep validation on a semantic level is to be ensured by the producer and consumer of CVRF documents. It is out of scope for this specification. Never the less proven and intended usage patterns from practice are given where possible.

Delegation to industry best practices technologies is used in referencing schemas for:

- **Document Metadata**:
  - Dublin Core (DC) Metadata Initiative Version 1.1 [DCMI11]
    - XML Namespace `http://purl.org/dc/elements/1.1/`
- **Platform Data**:
  - Common Platform Enumeration (CPE) Version 2.3 [CPE23_A]
    - XML Namespace `http://cpe.mitre.org/language/2.0`
- **Security Content Automation**:
  - Security Content Automation Protocol (SCAP) Version 1.2 [SCAP12]
    - XML Namespace `http://scap.nist.gov/schema/scap-core/1.0`
- **Vulnerability Scoring**:
  - Common Vulnerability Scoring System (CVSS) Version 3.0 [CVSS3]
    - XML Namespace `https://www.first.org/cvss/cvss-v3.0.xsd`
  - Common Vulnerability Scoring System (CVSS) Version 2.0 [CVSS2] [1]
    - XML Namespace `http://scap.nist.gov/schema/cvss-v2/1.0`

[1] Deprecated CVSS v2 provided for compatibility; some vendors switched to CVSS v3 in 2016 already.

# 3 CVRF Model Tree Map

To assist navigating the topology of the CSAF CVRF version 1.2 document schema, a graphical tree rendering of the parent-child-grandchild relations among the elements under the single `cvrf:cvrfdoc` root is provided in Figure 1:

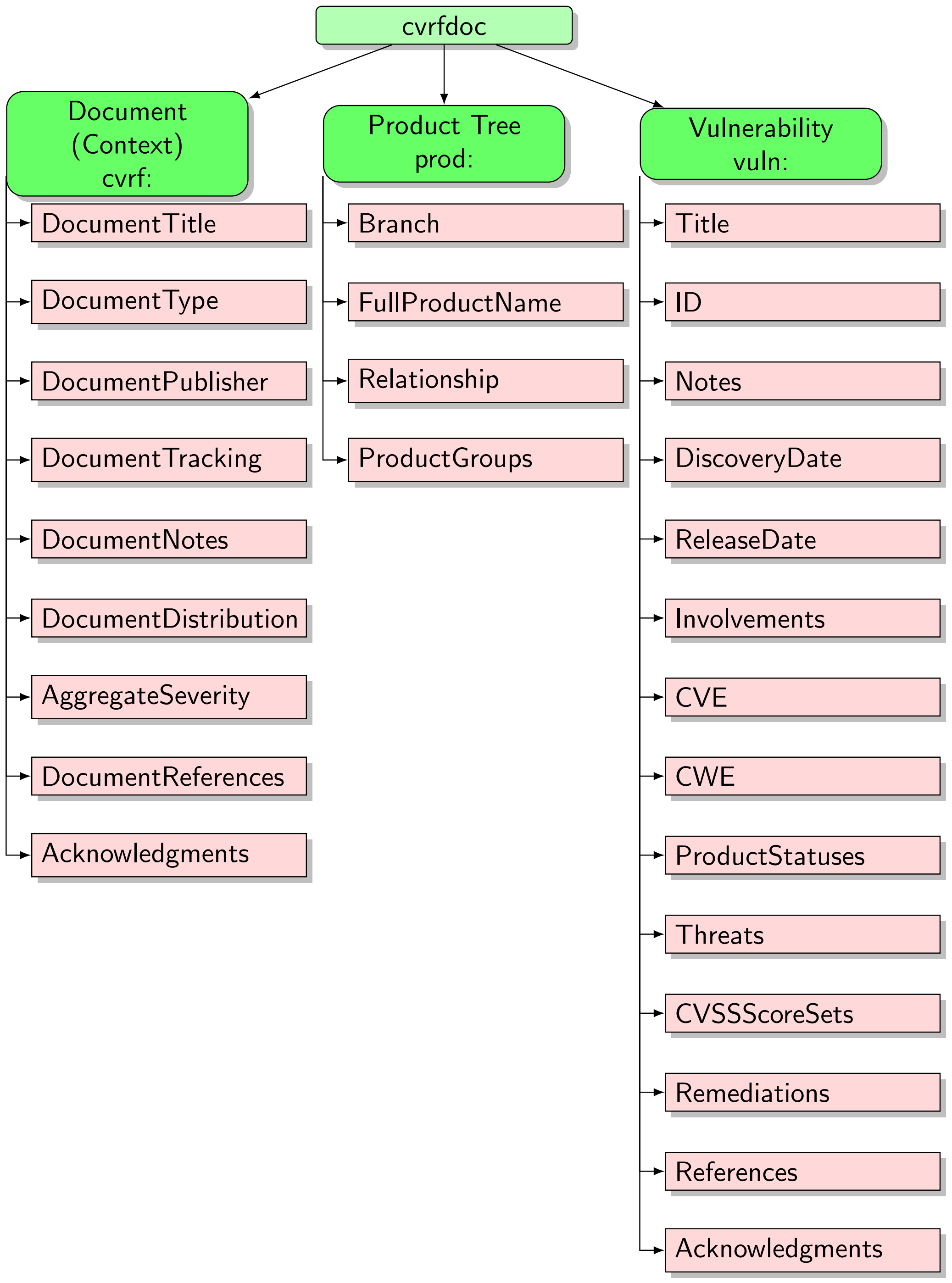

*Figure 1: First and second level sectioning below the `cvre:cvredoc` root.*

# 4 Document (Context) Schema Elements

The following nine top-level elements are defined in the `cvrf` XML schema file and if given must appear in the order listed and as children of the `cvrf:cvrfdoc` single root element:

| | | |
|---|---|---|
| 1. | Title: | `cvrf:DocumentTitle` |
| 2. | Type: | `cvrf:DocumentType` |
| 3. | Publisher: | `cvrf:DocumentPublisher` |
| 4. | Tracking: | `cvrf:DocumentTracking` |
| 5. | Notes: | `cvrf:DocumentNotes` |
| 6. | Distribution: | `cvrf:DocumentDistribution` |
| 7. | Aggregate Severity: | `cvrf:AggregateSeverity` |
| 8. | References: | `cvrf:DocumentReferences` |
| 9. | Acknowledgements: | `cvrf:Acknowledgements` |

The remaining sub sections will describe the elements, requirements on them and state recommendations and examples.

## 4.1 Document Title

**`cvrf:DocumentTitle`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |

The element `cvrf:DocumentTitle` holds a definitive canonical name for the document, providing enough descriptive content to differentiate from other similar documents, ideally providing a unique "handle" While this field is largely up to the document producer, common usage brings some recommendations:

The title should be succinct and promptly give the reader an idea of what is to come. If the document producer also publishes a human-friendly document that goes hand-in-hand with a CVRF document, it is recommended that both documents use the same title. It is further recommended to include the manufacturer name with any product names mentioned in the title.

### Examples

*Example 2:*

```
<DocumentTitle>Cisco IPv6 Crafted Packet Vulnerability</DocumentTitle>
```

*Example 3:*

```
<DocumentTitle>CERT Vulnerabilities in Kerberos 5 Implementation</DocumentTitle>
```

*Example 4:*

```
<DocumentTitle>Cisco Content Services Switch 11000 Series DNS Negative Cache of
Information  Denial-of-Service Vulnerability</DocumentTitle>
```

*Example 5:*

```
<DocumentTitle>Symantec Brightmail AntiSpam Static Database Password</DocumentTitle>
```

*Example 6:*

```
<DocumentTitle>HPSBUX02697 SSRT100591 rev.1 - HP-UX Running Java, Remote Unauthorized
Access, Disclosure of Information, and Other Vulnerabilities</DocumentTitle>
```

*Example 7:*

```
<DocumentTitle>Microsoft Vulnerability in the Microsoft Data Access Components (MDAC)
Function Could Allow Code Execution</DocumentTitle>
```

*Example 8:*

```
<DocumentTitle>
  Microsoft Vulnerability in Windows Explorer Could Allow Remote Code  Execution
</DocumentTitle>
```

## 4.2 Document Type

**cvrf:DocumentType**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |

The element `cvrf:DocumentType` defines a short canonical name, chosen by the document producer, which will inform the end user as to the type of document.

### Examples

*Example 9:*

```
<DocumentType>Vulnerability Report</DocumentType>
```

*Example 10:*

```
<DocumentType>Security Bulletin</DocumentType>
```

*Example 11:*

```
<DocumentType>Security Notice</DocumentType>
```

## 4.3 Document Publisher

**cvrf:DocumentPublisher**

| | |
|---|---|
| **Data Type:** | string |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Contact Details, Issuing Authority |
| **Attribute:** | Type, Vendor ID |
| **Attribute Data Type:** | enumerated list, string |
| **Attribute Range:** | {Vendor, Discoverer, Coordinator, User, Other}, unrestricted |
| **Attribute Required:** | yes, no |

The element `cvrf:DocumentPublisher` is a container that holds all the information about the publisher of the CVRF document, including attributes denoting the Type of publisher and an optional Vendor ID as well as optional elements for **Contact Details** and **Issuing Authority**.

**Document Publisher** is a required element, but the only required data is the *Type* attribute. The **Document Publisher** *Type* attribute is an enumerated list containing an array of different document publisher types. Types include:

- **Vendor**: Developers or maintainers of information system products or services. This includes all authoritative product vendors, Product Security Incident Response Teams (PSIRTs), and product resellers and distributors, including authoritative vendor partners.

- **Discoverer**: Individuals or organizations that find vulnerabilities or security weaknesses. This includes all manner of researchers.
- **Coordinator**: Individuals or organizations that manage a single vendor's response or multiple vendors' responses to a vulnerability, a security flaw, or an incident. This includes all Computer Emergency/Incident Response Teams (CERTs/CIRTs) or agents acting on the behalf of a researcher.
- **User**: Everyone using a vendor's product.
- **Other**: Catchall for everyone else. Currently this includes forwarders, republishers, language translators, and miscellaneous contributors.

The optional *Vendor ID* attribute is a unique identifier (OID) that a vendor uses as issued by FIRST under the auspices of IETF. At the time of this writing, OID is a work in progress.

#### Example

*Example 12:*

```
<DocumentPublisher Type="Vendor" VendorID="MarcusCom"/>
```

### 4.3.1 Document Publisher – Contact Details

**`cvrf:DocumentPublisher / cvrf:ContactDetails`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Document Publisher |

The optional Document Publisher element `cvrf:ContactDetails` contains information required to contact the document publisher.

#### Example

*Example 13:*

```
<ContactDetails>
  Name: Captain Sledge Fisthammer\r\nOrganization: International Space Explorers
  of America\r\nPhone Number: 555-123-4567\r\nFax Number: 555-123-4568\r\n
  Email Address: sledge@foo.com
</ContactDetails>
```

### 4.3.2 Document Publisher – Issuing Authority

**`cvrf:DocumentPublisher / cvrf:IssuingAuthority`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Document Publisher |

The optional Document Publisher element `cvrf:IssuingAuthority` states the name of the issuing party and that party's authority to release the document. In particular, it addresses the party's constituency and responsibilities or other obligations. This element should also include instructions for contacting the issuer.

### Example

*Example 14:*

```
<IssuingAuthority>
  The Juniper SIRT (Juniper Networks Security Incident Response Team) is the sole
  authority regarding vulnerabilities in any Juniper Networks products or services,
  and coordinates the handling of all aspects of such vulnerabilities from initial
  discovery or report through public announcements and any subsequent follow-on
  activities. Additional information is available at
  http://www.juniper.net/support/security/report_vulnerability.html
</IssuingAuthority>
```

## 4.4 Document Tracking

**`cvrf:DocumentTracking`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 (required) |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Identification, Status, Version, Revision History, Initial Release Date, Current Release Date, Generator |

The element `cvrf:DocumentTracking` is a container deisgnated to hold all management attributes necessary to track a CVRF document as a whole.

### Example

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

### 4.4.1 Document Tracking – Identification

**`cvrf:DocumentTracking / cvrf:Identification`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Document Tracking |

The Document Tracking element `cvrf:Identification` is a container that holds all the identifiers for the CVRF document. Required is the **ID** element, optional is the **Alias** element.

### Example

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

#### 4.4.1.1 Document Tracking – Identification – ID

**cvrf:DocumentTracking / cvrf:Identification / cvrf:ID**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Identification |

The element `cvrf:ID` is a short, unique identifier used to refer to the document unambiguously in any context. The ID is a simple label. It is a string data type to provide for a wide range of numbering values, types, and schemes. Typically, the ID should be assigned and maintained by the original document issuing authority. It is recommended that the ID be a monotonically increasing value, or increasing in such a predictable manner that it does not contribute toward confusion or misnumbering. Careful consideration is required to ensure that construction of the ID does not contribute to confusion or collision with other labels.

**Example**

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

#### 4.4.1.2 Document Tracking – Identification – Alias

**cvrf:DocumentTracking / cvrf:Identification / cvrf:Alias**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Identification |

The optional element `cvrf:Alias` is an optional alternative ID used to refer to the document. Many vendors have one or more alternative or secondary IDs for documents and the **Alias** presents an interface to publish those alongside the primary ID.

**Example**

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

### 4.4.2 Document Tracking – Status

**cvrf:DocumentTracking / cvrf:Status**

| | |
|---|---|
| **Data Type:** | enumerated list |
| **Range:** | {Draft, Interim, Final} |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Document Tracking |

The element `cvrf:Status` content refers to the condition of the document with regard to completeness and the likelihood of future editions.

Status types are:

- **Draft**: Pre-release, intended for issuing party's internal use only, or possibly used externally when the party is seeking feedback or indicating its intentions regarding a specific issue.
- **Interim**: The issuing party believes the content is subject to change.
- **Final**: The issuing party asserts the content is unlikely to change. "Final" status is an indication only, and does not preclude updates.

Issuing parties are strongly recommended to set Status to "Draft" when initiating a new document and to implement procedures to ensure that the status is changed to the appropriate value before the document is released.

#### Examples

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

### 4.4.3 Document Tracking – Version

**`cvrf:DocumentTracking / cvrf:Version`**

| | |
|---|---|
| **Data Type:** | token |
| **Range:** | unrestricted.unrestricted.unrestricted.unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Document Tracking |

The element `cvrf:Version` holds a simple counter to track the version of the document. This is a numeric tokenized field of the format "`nn`" – "`nn.nn.nn.nn`". It may be incremented in either major or minor notation to denote clearly the evolution of the content of the document. Issuing parties must ensure that this field is incremented appropriately, even for the least editorial or grammatical changes, when the field is used. It is validated using the following regular expression:

```
(0|[1- 9][0-9]*)(\.(0|[1-9][0-9]*)){0,3}
```

#### Examples

*Example 15:*

```
<Version>1.0</Version>
```

*Example 16:*

```
<Version>1.0.1</Version>
```

*Example 17:*

```
<Version>1.0.0.1</Version>
```

### 4.4.4 Document Tracking – Revision History

**cvrf:DocumentTracking / cvrf:RevisionHistory**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Document Tracking |
| **Children:** | Revision |

The element `cvrf:RevisionHistory` is a container, that should contain one **Revision** entry for each version/revision of the document, including the initial version and entries for each subsequent update.

#### Example

```
See example in section 4.4.4.1.3
```

#### 4.4.4.1 Document Tracking – Revision History – Revision

**cvrf:DocumentTracking / cvrf:RevisionHistory / cvrf:Revision**

| | |
|---|---|
| **Data Type:** | container |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Revision History |
| **Children:** | Number, Date, Description |

The element `cvrf:Revision` contains all the elements required to track the evolution of a CVRF document. Each change to a CVRF document should be accompanied by **Number**, **Date**, and **Description** elements.

#### Example

```
See example in section 4.4.4.1.3
```

##### 4.4.4.1.1 Document Tracking – Revision History – Revision – Number

**cvrf:DocumentTracking / ... / cvrf:Revision / cvrf:Number**

| | |
|---|---|
| **Data Type:** | token |
| **Range:** | unrestricted.unrestricted.unrestricted.unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Revision |

The element `cvrf:Number` should contain the numeric version of the document. Like the **Version** element above, it is a numeric tokenized field of the format “`nn`” with up to four fields “`nn.nn.nn.nn`”. It is recommended that this be a monotonically increasing value. Minor revisions should be used for less-significant changes (for example, `1.0.0.0` to `1.0.0.1`). Major, actionable changes should lead to a major increase of the version number (for example, `1.0` to `2.0`).

Examples of such changes include:

- Any change to severity or impact
- The announcement of additional vulnerabilities
- The announcement of additional vulnerable products
- A significant change in remediation status

The most recent **Number** element should *always* match the **Version** element. It is validated using the following regular expression:

```
(0|[1- 9][0-9]*)(\.(0|[1-9][0-9]*)){0,3}
```

### Example

```
See example in section 4.4.4.1.3
```

## 4.4.4.1.2 Document Tracking – Revision History – Revision – Date

**`cvrf:DocumentTracking / ... / cvrf:Revision / cvrf:Date`**

| | |
|---|---|
| **Data Type:** | dateTime |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Revision |

The element `cvrf:Date` should record the date the revision was made. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Greenwich mean time [GMT] or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

### Example

```
See example in section 4.4.4.1.3
```

## 4.4.4.1.3 Document Tracking – Revision History – Revision – Description

**`cvrf:DocumentTracking / ... / cvrf:Revision / cvrf:Number`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Revision |

The element `cvrf:Description` should be a short description of the changes made. It can describe the conditions that prompted the change or be a short list of the items changed.

### Example

*Example 18:*

```
<RevisionHistory>
  <Revision>
```

```
    <Number>1</Number>
    <Date>2011-11-26T00:00:00+00:00</Date>
    <Description>initial public release</Description>
  </Revision>
</RevisionHistory>
```

### 4.4.5 Document Tracking – Initial Release Date

**cvrf:DocumentTracking / cvrf:InitialReleaseDate**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Document Tracking |

The element `cvrf:InitialReleaseDate` is the date (and time, optionally) that the document was initially released by the issuing party. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Universal Time Coordinated (UTC), also known as GMT or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

#### Examples

*Example 19:*

```
<InitialReleaseDate>2011-11-26T00:00:00+00:00</InitialReleaseDate>
```

### 4.4.6 Document Tracking – Current Release Date

**cvrf:DocumentTracking / cvrf:CurrentReleaseDate**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Document Tracking |

The element `cvrf:CurrentReleaseDate` is the current date (and time, optionally) that the document was released by the issuing party. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Universal Time Coordinated (UTC), also known as GMT or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

#### Examples

*Example 20:*

```
<CurrentReleaseDate>2011-11-26T00:00:00+00:00</CurrentReleaseDate>
```

## 4.4.7 Document Tracking – Generator

**`cvrf:DocumentTracking / cvrf:Generator`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Document Tracking |
| **Children:** | Engine, Date |

The element `cvrf:Generator` is a container to hold all elements related to the generation of the document. These items will reference when the document was actually created, including the date it was generated and the entity that generated it.

### Example

```
See example in section 4.4.7.1.2
```

#### 4.4.7.1.1 Document Tracking – Revision History – Revision – Engine

**`cvrf:DocumentTracking / ... / cvrf:Revision / cvrf:Engine`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Revision |

The optional element `cvrf:Engine` should refer to the name and optional version of the engine that generated the CVRF document.

### Example

```
See example in section 4.4.7.1.2
```

#### 4.4.7.1.2 Document Tracking – Revision History – Revision – Date

**`cvrf:DocumentTracking / ... / cvrf:Revision / cvrf:Date`**

| | |
|---|---|
| **Data Type:** | dateTime |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Revision |

The optional element `cvrf:Date` should refer to the date the CVRF document was generated. Because documents are often generated internally by a document producer and exist for a nonzero amount of time before being released, this field can be different from the **Initial Release Date**. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Universal Time Coordinated (UTC), also known as GMT or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

#### Example

*Example 21:*

```
<Generator>
  <Engine>Mike Schiffman’s sublime fingertips version 1.0</Engine>
  <Date>2012-02-27T00:00:00+00:00</Date>
</Generator>
```

## 4.5 Document Notes

**`cvrf:DocumentNotes`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Note |

The optional element `cvrf:DocumentNotes` is a container that holds all of the document-level **Note** elements.

#### Example

```
See example in section 4.5.1
```

### 4.5.1 Document Notes – Note

**`cvrf:DocumentNotes / cvrf:Note`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Identification |

The element `cvrf:Note` is a place to put all manner of text blobs related to the document as a whole. It can be a concise summary of the overall document or a more compartmentalized and area-specific textual discussion. Depending on the need, there can be zero, one, or several **Note** elements in a given CVRF document.

The note should contain a compartmentalized textual discussion constrained by its *Type* attribute. *Type* can be one of the following:

- **General**: A general, high-level note (Title may have more information).
- **Details**: A low-level detailed discussion (Title may have more information).
- **Description**: A description of something (Title may have more information).
- **Summary**: A summary of something (Title may have more information).
- **FAQ**: A list of frequently asked questions.
- **Legal Disclaimer**: Any possible legal discussion, including constraints, surrounding the document.
- **Other**: Something that doesn’t fit (Title should have more information).

Title and Audience are optional attributes to give human readers context around what they are about to read; Title should be a concise description of what is contained in the text, whereas Audience will indicate who is intended to read it.

For example, when *Type* is “General”, *Title* is “executive summary”, and *Audience* is “executives”, the note is a high-level overview designed for consumption by C-level decision makers. It should be brief and devoid of any technical details and jargon. On the other hand, when *Type* is “Details”, *Title* is “technical summary”, and *Audience* is “operational management and system administrators”, the note will be more detailed in nature and will contain more operational information.

Ordinal is a mandatory, locally significant value used to track notes inside a CVRF document at the root (document) level. It is provided to uniquely identify a Note. There should be one of these values for every Note inside Document Notes, and it is recommended that Ordinal should be instantiated as a monotonically increasing counter, indexed from 1. Each Ordinal that tracks a Note inside Document Notes is completely independent from an Ordinal tracking a Note inside Vulnerability/Notes.

### Example

*Example 22:*

```
<DocumentNotes>
  <Note Type="General" Ordinal="1" Title="Details" Audience="All">
    These are some details about a CVRF document intended for all stakeholders.
  </Note>
</DocumentNotes>
```

## 4.6 Document Distribution

**`cvrf:DocumentDistribution`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |

The optional element `cvrf:DocumentDistribution` should contain details about constraints, if any, for sharing the CVRF document with additional recipients. Constraints may include instructions on how to reproduce, share, copy, or otherwise distribute the document.

### Example

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

## 4.7 Aggregate Severity

**`cvrf:AggregateSeverity`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Contact Details, Issuing Authority |
| **Attribute:** | Namespace |
| **Attribute Data Type:** | anyURI |
| **Attribute Range:** | unrestricted |
| **Attribute Required:** | no |

The optional element `cvrf:DocumentXYZ` is a container that is provided by the document producer to convey the urgency and criticality with which the vulnerability or vulnerabilities should be addressed. It is a document-level metric and applied to the document as a whole—not any specific vulnerability. The range of values in this field is defined according to the document producer's policies and procedures. These values can be understood only in the context of the document producer's stated practices. Therefore, the values may vary widely depending on the source of the document. The field is independent of—and in addition to—any other standard metric for determining the impact or severity of a given vulnerability (such as CVSS).

If one exists, the attribute Namespace should contain a URL pointing to the namespace so referenced.

### Example

```
See example in section TODO_DEFINE_OR_REMOVE_NO_CURRENT_EXAMPLE_PROVIDED
```

## 4.8 Document References

**`cvrf:DocumentReferences`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Reference |

The optional element `cvrf:DocumentReferences` is a container that should include references to any conferences, papers, advisories, and other resources that are related and considered to be of value to the document consumer. For every **Document References** container, there must be at least one **Reference** element, and each **Reference** element must contain one **URL** and one **Description.**

### Example

```
See example in section 4.8.1.2
```

### 4.8.1 Document References – Reference

**cvrf:DocumentReferences / cvrf:Reference**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Document References |
| **Children:** | URL, Description |
| **Attribute:** | Type |
| **Attribute Data Type:** | enumerated list |
| **Attribute Required:** | yes |
| **Attribute Default Value:** | External |

The element `cvrf:Reference` refers to resources related to the overall CVRF document. These may include a plaintext or HTML version of the advisory or other related documentation, such as white papers or mitigation documentation.

The *Type* attribute denotes the type of the document reference relative to the given document. The follow types are available:

- **External:** The default value indicates the reference is external to the document.
- **Self**: This indicates the related document is actually a direct reference to itself

#### Example

```
See example in section 4.8.1.2
```

#### 4.8.1.1 Document References – Reference – URL

**cvrf:DocumentReferences / cvrf:Reference / cvrf:URL**

| | |
|---|---|
| **Data Type:** | anyURI |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Reference |

The element `cvrf:URL` in this container is the fixed URL or location of the reference.

#### Example

```
See example in section 4.8.1.2
```

#### 4.8.1.2 Document References – Reference – Description

**cvrf:DocumentReferences / cvrf:Reference / cvrf:Description**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Reference |

The element `cvrf:Description` holds a descriptive title or the name of the reference.

#### Example

*Example 23:*

```
<References>
  <Reference Type="External">
    <URL>http://foo.foo/bar/</URL>
    <Description xml:lang="fr">C'est un test de référence</Description>
  </Reference>
</References>
```

## 4.9 Acknowledgements

**`cvrf:Acknowledgements`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Acknowledgement |

The optional element `cvrf:Acknowledgements` is a container holds one or more **Acknowledgment** containers, which contain recognition of external parties.

#### Example

```
See example in section 4.9.1.4
```

### 4.9.1 Acknowledgements – Acknowledgement

**`cvrf:Acknowledgements / cvrf:Acknowledgement`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgements |
| **Children:** | Name, Organization, Description, URL |

The element `cvrf:Acknowledgement` contains recognition of external parties that reported noncritical/low- severity security issues or provided information, observations, or suggestions that contributed to improved security or improved documentation in future releases of the document producer's products. This may also contain recognition to external parties that contributed toward producing this document.

An acknowledgment container may contain four different types of child elements: **Name**, **Organization**, **Description**, and **URL**. All are described in the subsections below.

#### Example

```
See example in section 4.9.1.4
```

### 4.9.1.1 Acknowledgements – Acknowledgement – Name

**`cvrf:Acknowledgements / cvrf:Acknowledgement / cvrf:Name`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgement |

The element `cvrf:Name` should contain the name of the party being acknowledged.

#### Example

```
See example in section 4.9.1.4
```

### 4.9.1.2 Acknowledgements – Acknowledgement – Organization

**`cvrf:Acknowledgements / cvrf:Acknowledgement / cvrf:Organization`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Acknowledgement |

The element `cvrf:Organization` should contain the organization of the party or if the **Name** is omitted, the organization itself that is being acknowledged.

#### Example

```
See example in section 4.9.1.4
```

### 4.9.1.3 Acknowledgements – Acknowledgement – Description

**`cvrf:Acknowledgements / cvrf:Acknowledgement / cvrf:Description`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgement |

The element `cvrf:Description` can contain any contextual details the document producers wish to make known about the acknowledgment or acknowledged parties.

If attributing to multiple organizations, each contributor should be grouped with that **Organization** within a single **Acknowledgment** container. An **Organization**-specific acknowledgment may be added within each **Acknowledgment** container using the **Description** element. If an overall general or aggregate acknowledgment is to be added, an **Acknowledgment** container that contains a single **Description** element may be used.

#### Example

```
See example in section 4.9.1.4
```

#### 4.9.1.4 Acknowledgements – Acknowledgement – URL

**`cvrf:Acknowledgements / cvrf:Acknowledgement / cvrf:URL`**

| | |
|---|---|
| **Data Type:** | anyURI |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgement |

The element `cvrf:URL` is is the optional URL to the person, place, or thing being acknowledged.

#### Example

*Example 24:*

```
<Acknowledgments>
  <Acknowledgment>
    <Name>Bartholomew McHandsome</Name>
    <Organization>FancyPants Inc.</Organization>
    <Description>We couldn’t have done it without you Bart!</Description>
    <URL>http://foo.foo/bar/</URL>
  </Acknowledgment>
</Acknowledgments>
```

# 5 Product Tree Schema Elements

Product information in CVRF is modeled as zero or one top-level Product Tree element instance of `prod:ProductTree` (defined in the product tree schema file within the `prod` namespace).

The following 4 second-level elements are and must appear in the order listed if given as elements of the top-level element Product Tree:

| | | |
|---|---|---|
| 1. | Branch: | `prod:Branch` |
| 2. | Full Product Name | `prod:FullProductName` |
| 3. | Relationship: | `prod:Relatinship` |
| 4. | Product Groups: | `prod:ProductGroups` |

The remaining sub sections will describe the above 5 first and second level elements together with their children and grandchildren, constraints on them as well as state recommendations and examples.

To avoid duplication of data and accommodate for the many possible complex relationships among real world products, the 4 above named elements maybe nested deeply (e.g. a **Branch** of a **Branch** ...) or are clearly useful in many places like the **Full Product Name**.

The sub sections introducing **Branch, Relationship**, and **Product Groups** try to further offer such topological usage info to aid the reader in creating or navigating the graph that can be spanned by instances of a **Product Tree**.

As a service to the reader otherwise commonly used acronyms used in the rest of this section (or found in the XML schema source files are expanded here and as understood in the context of CVRF (for more details please cf. the respective sub sections below):

- CPE: Common Platform Enumeration (cf. [CPE23-N], [CPE23-M], [CPE23-D], and [CPE23-A])
- CVSS: Common Vulnerability Scoring System (cf. [CVSS3] preferred, [CVSS2] deprecated)
- SCAP: Security Content Automation Protocol (cf. [SCAP12])

## 5.1 Product Tree

**`prod:ProductTree`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Root |
| **Children:** | Branch, Full Product Name, Relationship |

The optional element `cvrf:DocumentReferences` is a container for all fully qualified product names that can be referenced elsewhere in the document (specifically when describing the products that are affected by a vulnerability using the **Product Statuses**, **Threats**, **CVSS Score Sets**, and **Remediation** containers). The **Product Tree** can have as many branches as needed, but each endpoint of the tree must be terminated with a **Full Product Name** element, which represents a product that can be referenced elsewhere.

This structure is a major change from CVRF 1.0, where affected products were direct child elements of **Vulnerability**; the change was necessary to meet the vastly different requirements of various organizations in the way they document their products. Also, in situations where a CVRF document contains more than one vulnerability, a separate product repository at the document level reduces the need to duplicate all product entries in each vulnerability.

The **Product Tree** can be kept simple (flat) or made more detailed (branched out). It also supports concatenating products to describe relationships, such as components contained in a product or products installed on other products.

**Flat:**
In the simplest case, a flat **Product Tree** would contain one or more **Full Product Name** elements at the root level, one for each product that needs to be described.

**Branched:**
In a more detailed **Product Tree**, the root element would contain one or more **Branch** elements at the root level, one for each class/type/category of product, each of which again contains one or more **Branch** elements until all desired categories and subcategories are described to the satisfaction of the document issuer. Then each open **Branch** element is terminated with the actual product item in the form of a **Full Product Name** element.

**Concatenated:**
No matter whether a flat or branched structure is chosen, you may need to be able to describe the combination of two **Full Product Name** elements, such as when a product is only vulnerable when installed together with another, or to describe operating system components. To do that, a **Relationship** element is inserted at the root of the **Product Tree**, with attributes establishing a link between two existing **Full Product Name** elements, allowing the document producer to define a combination of two products that form a new **Full Product Name** entry.

**Grouped:**
Once **Full Product Name** elements are defined, they may be freely added to logical groups, which may then be used to refer to a group of products. Given that it is possible for a product to be a member of more than one logical group, some areas of the CVRF document may not allow references to product groups to avoid ambiguity.

### Example

*Example 25:*

```
<prod:ProductTree>
  <prod:Branch Name="Vendorix" Type="Vendor">
    <prod:Branch Name="... Appliances" Type="Product Name">
      <prod:Branch Name="1.0" Type="Product Version">
        <prod:Branch Name=".0" Type="Service Pack">
          <prod:FullProductName ProductID="CVRFPID-223152">
            ... AppY 1.0.0
          </prod:FullProductName>
        </prod:Branch>
        <prod:Branch Name="(2)" Type="Service Pack">
          <prod:FullProductName ProductID="CVRFPID-223153">
            ... AppY 1.0(2)
          </prod:FullProductName>
        </prod:Branch>
      </prod:Branch>
      <prod:Branch Name="1.1" Type="Product Version">
        <prod:Branch Name=".0" Type="Service Pack">
          <prod:FullProductName ProductID="CVRFPID-223155">
            ... AppY 1.1.0
          </prod:FullProductName>
        </prod:Branch>
        <prod:Branch Name="(1)" Type="Service Pack">
          <prod:FullProductName ProductID="CVRFPID-223156">
            ... AppY 1.1(1)
          </prod:FullProductName>
        </prod:Branch>
      </prod:Branch>
    </prod:Branch>
  </prod:Branch>
</prod:ProductTree>
```

### Visual Overview

Map of **Product Tree** including the parent node (**Document**) in some valid configuration spanning multiple sub trees:

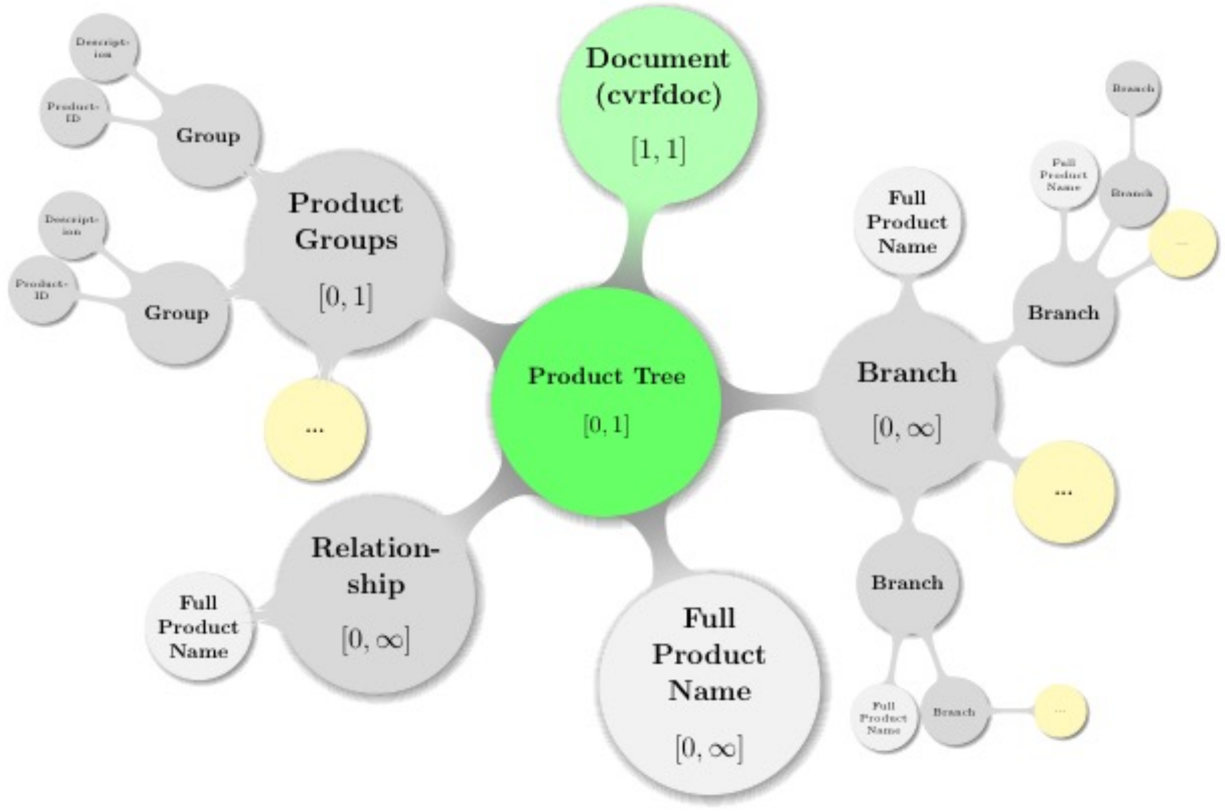

*Figure 2: Visual presentation of abstract but topologically valid Product Tree instance.*

Some decent coloring has been applied to above graph to balance visual hints with accessibility. The mathematical closed interval notation has been used to annotate the minimum and maximum occurrences of elements, where the infinity symbol (∞) translates to the term unbounded in XML lingo.

The pale gray color of the **Full Product Name** representative nodes shall indicate that they are more used like labels all over the topology. The nodes carrying an ellipsis (…) shall hint at possible further deep nesting of the sub trees where they are attached.

## 5.1.1 Product Tree – Branch

**`prod:ProductTree / prod:Branch`**

| | |
|---|---|
| **Data Type:** | choice |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Product Tree, Branch |
| **Children:** | Branch, Full Product Name |
| **Attribute:** | Name, Type |
| **Attribute Data Type:** | string, enumerated list |
| **Attribute Required:** | yes, yes |

The `cvrf:Branch` element is a choice element. A choice element behaves as a regular container that can have different child elements, but with the difference that only exactly one child element can be chosen. It is similar in concept to the “union” programming construct in which one variable can have one of several different predefined data types.

The **Branch** element contains a *Type-Name* pair as mandatory attributes to describe the characteristics of the current **Branch**. *Type* will describe the category of the branch in question. A full universe of values for version 1.2 is shown below.

Branch *Types*:

- **Vendor**: The name of the vendor or manufacturer that makes the product
- **Product Family**: The product family that the product falls into
- **Product Name**: The name of the product
- **Product Version**: The product version, can be numeric or some other descriptor
- **Patch Level:** The patch level of the product
- **Service Pack**: The service pack of the product
- **Architecture**: The architecture for which the product is intended
- **Language**: The language of the product
- **Legacy**: A nonspecific legacy entry
- **Specification**: A specification such as a standard, best common practice, etc.

*Name* will contain the canonical descriptor or "friendly name" of the branch.

As for the child elements, each **Branch** can have either one of the following children:

- One **Full Product Name**. A single child element terminates the branch by describing a final product entry (described below).
- More **Branches**. Multiple additional **Branch** containers, which on their own can either terminate in a single **Full Product Name** element or yet more **Branch** containers.

### Example

*Example 26: Nesting of **Branches** in a **Branch** subtree:*

```
<prod:Branch Type="Vendor" Name="Microsoft">
  <prod:Branch Type="Product Family" Name="Windows">
    <prod:Branch Type="Product Name" Name="Vista">
      <prod:Branch Type="Service Pack" Name="1">
        <prod:FullProductName ProductID="CVRFPID-0001">
          Microsoft Windows Vista Service Pack 1
        </prod:FullProductName>
      </prod:Branch>
      <prod:Branch Type="Service Pack" Name="2">
        <prod:FullProductName ProductID="CVRFPID-0002">
          Microsoft Windows Vista Service Pack 2
        </prod:FullProductName>
      </prod:Branch>
    </prod:Branch>
  </prod:Branch>
  <prod:Branch Type="Product Family" Name="Office">
    <prod:Branch Type="Product Name" Name="Word 2010">
      <prod:Branch Type="Service Pack" Name="0">
        <prod:Branch Type="Architecture" Name="x86">
          <prod:FullProductName ProductID="CVRFPID-0003">
            Microsoft Word 2010 (32-bit editions)
          </prod:FullProductName>
        </prod:Branch>
      </prod:Branch>
    </prod:Branch>
  </prod:Branch>
</prod:Branch>
```

A more visual display of the same structure from above example is shown in the figure below (Figure 3: Graphical display of a Product Tree - Branch combination).

### Visual Overview

Map of **Branch** sub tree from above example of nested Branches  including the parent node (**Product Tree** left out in XML source code example) with some textual hints to map the topologies:

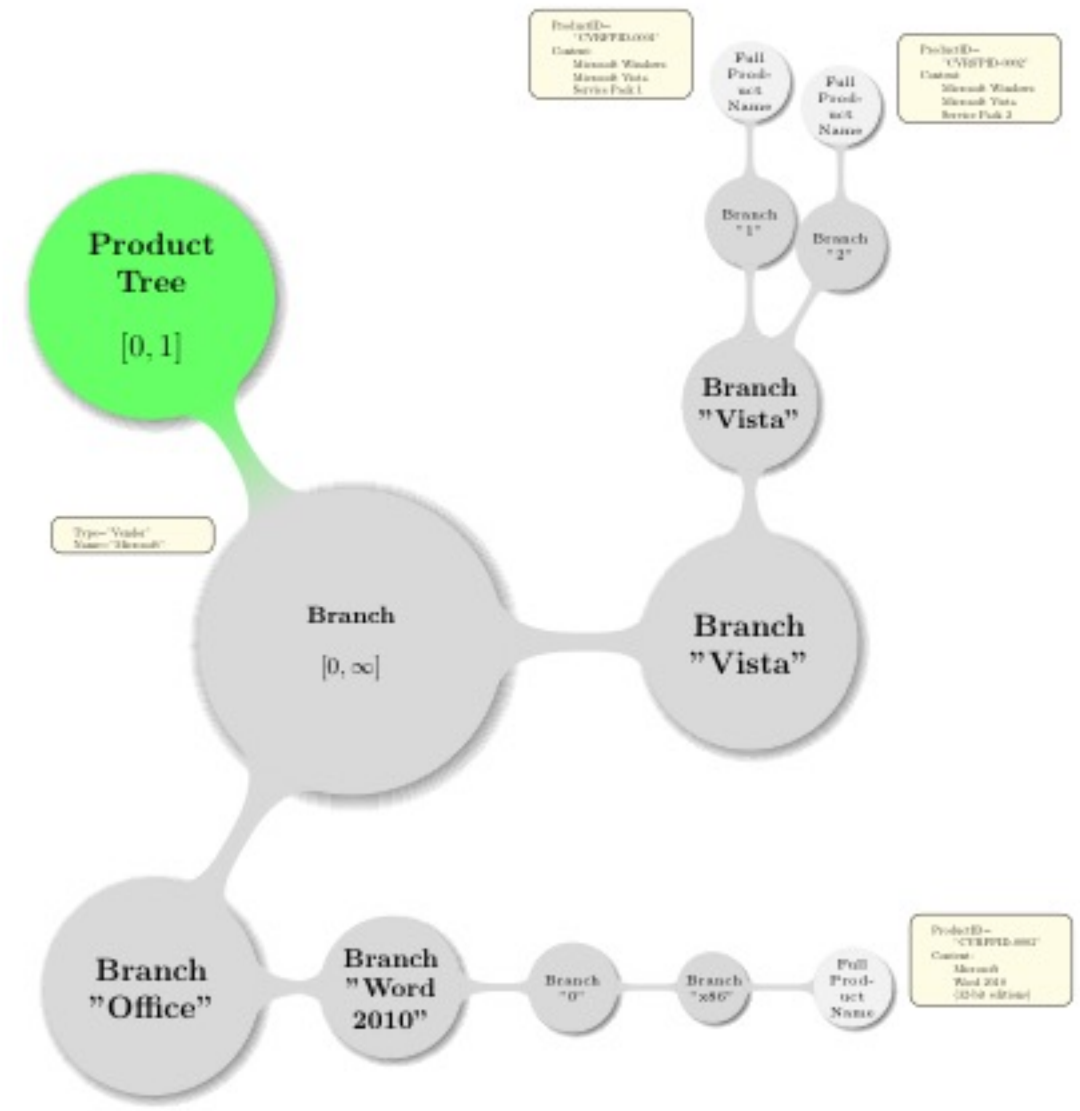

*Figure 3: Graphical display of a Product Tree - Branch combination*

## 5.1.2 Product Tree – Full Product Name

**prod:ProductTree / prod:FullProductName**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Product Tree, Relationship, Branch |
| **Attribute:** | Product ID, CPE |
| **Attribute Data Type:** | token, CPE number or CPE URI |
| **Attribute Required:** | yes, no |

The `cvrf:FullProductName` elements define the endpoints of the **Product Tree** and occur directly at the root level, at the branch level, or as the result of a relationship between two products. The value of a **Full Product Name** element should be the product's full canonical name, including version number and other attributes, as it would be used in a human-friendly document.

The Common Platform Enumeration (*CPE*) attribute refers to a method for naming platforms. The structure for CPE is described at http://cpe.mitre.org. The *CPE* can be either an integer (if MITRE has an entry for the platform in question) or a candidate string from the vendor if no MITRE entry yet exists.

The *Product ID* attribute is required to identify a **Full Product Name** so that it can be referred to from other parts in the document. There is no predefined or required format for the *Product ID* as long as it uniquely identifies a product in the context of the current document. Examples include incremental integers or Globally Unique Identifiers (GUIDs).

#### Examples

*Example 27:*

```
<FullProductName ProductID="CVRFPID-0004">
  Microsoft Host Integration Server 2006 Service Pack 1
</FullProductName>
```

*Example 28:*

```
<FullProductName ProductID="CVRFPID-0005">
  Microsoft Office 2008 for Mac 12.3.1 Update
</FullProductName>
```

### 5.1.3 Product Tree – Relationship

**`prod:ProductTree / prod:Relationship`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Product Tree |
| **Children:** | Full Product Name |
| **Attribute:** | Product Reference, Relationship Type, Relates To Product Reference |
| **Attribute Data Type:** | token, enumerated list, token |
| **Attribute Required:** | yes, yes, yes |

The `cvrf:Relationship` element establishes a link between two existing **Full Product Name** elements, allowing the document producer to define a combination of two products that form a new **Full Product Name** entry.

This situation arises when a product is vulnerable only when installed together with another, or to describe operating system components. As a **Relationship** connects two existing products with each other, there need to be at least two **Full Product Name** entries present in the **Product Tree** before a Relationship element can be created.

**Relationship** elements live at the root of a **Product Tree**, and they have three mandatory attributes: *Product Reference* and *Relates To Product Reference* each contain the *Product ID* token for the two products that will form the relationship, and the *Type* attribute defines how the products are related.

Consider two previously constructed products with *Product IDs* CVRFPID-0001 and CVRFPID- 0002. CVRF v1.1 supports the following **Relationship** *Type* values:

- **Default Component Of**: CVRFPID-0001 is a default component of CVRFPID-0002
- **Optional Component Of**: CVRFPID-0001 is an optional component of CVRFPID-0002
- **External Component Of**: CVRFPID-0001 is an external component of CVRFPID-0002

- **Installed On**: CVRFPID-0001 is installed on CVRFPID-0002
- **Installed With**: CVRFPID-0001 is installed with CVRFPID-0002   Once a **Relationship** element has been created, it needs to be completed by adding one **Full Product Name** element as a child, typically using a combination of the two related product names as a value.

## Examples

*Example 29:*

The first product is defined as:

```
<FullProductName ProductID="CVRFPID-0007">
  Active Directory Lightweight Directory Service
</FullProductName>
```

And the second product is defined as:

```
<FullProductName ProductID="CVRFPID-0008">
  Windows Vista Service Pack 2
</FullProductName>
```

And the relationship can then be defined as:

```
<Relationship ProductReference="CVRFPID-0007" RelationType="OptionalComponentOf"
 RelatesToProductReference = "CVRFPID-0008">
  <FullProductName ProductID=”CVRFPID-0009>
    Active Directory Lightweight Directory Service as an optional component of
    Windows Vista Service Pack 2
  </FullProductName>
</Relationship>
```

*Example 30:*

In another example, the first product is defined as:

```
<FullProductName ProductID="CVRFPID-0010">
  Cisco AnyConnect Secure Mobility Client 2.3.185
</FullProductName>
```

And the second product is defined as:

```
<FullProductName ProductID="CVRFPID-0011">Microsoft Windows</FullProductName>
```

And the relationship can then be defined as:

```
<Relationship ProductReference="CVRFPID-0010" RelationType="InstalledOn"
 RelatesToProductReference="CVRFPID-0011">
  <FullProductName ProductID=”CVRFPID-0012>
    Cisco AnyConnect Secure Mobility Client 2.3.185 when installed on Microsoft Windows
  </FullProductName>
</Relationship>
```

### 5.1.4 Product Tree – Product Groups

**prod:ProductTree / prod:ProductGroups**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Product Tree |
| **Children:** | Group |

The optional element `cvrf:ProductGroups` is a container, that defines whether **Full Product Name** elements in the product tree will be grouped into logical groups. If the container is present, at least one Group must be defined.

If groups are defined, products can be referred to using the Group ID attribute in many other parts of the document, rather than repeatedly having to list all members individually.

Whether groups are defined or not, the ability to reference each product individually in other parts of the document is not affected. In fact, the creator of a document can choose to use either direct product references or group references.

Note:

Given that a single product can be a member of more than one group, some areas of the CVRF document may not allow product references by group to avoid ambiguity.

#### Example

*Example 31:*

We create two groups, `CVRFGID-0001` and `CVRFGID-0002`. Both groups have four members, and `ProductID` `CVRFPID-0001` is a member of both groups:

```
<ProductGroups>
  <Group GroupID="CVRFGID-0001">
    <ProductID>CVRFPID-0001</ProductID>
    <ProductID>CVRFPID-0002</ProductID>
    <ProductID>CVRFPID-0003</ProductID>
    <ProductID>CVRFPID-0004</ProductID>
  </Group>
  <Group GroupID="CVRFGID-0002">
    <ProductID>CVRFPID-0001</ProductID>
    <ProductID>CVRFPID-0010</ProductID>
    <ProductID>CVRFPID-0011</ProductID>
    <ProductID>CVRFPID-0099</ProductID>
  </Group>
</ProductGroups>
```

### 5.1.4.1 Product Tree – Product Groups – Group

**prod:ProductTree / prod:ProductGroups / prod:Group**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Product Groups |
| **Children:** | Description, Product ID |
| **Attribute:** | Group ID |
| **Attribute Data Type:** | token |
| **Attribute Required:** | yes |

The element `cvrf:Group` is a container, that defines a new logical group of products that can then be referred to in other parts of the document to address a group of products with a single identifier. **Group** members are defined by adding one **Product ID** element for each member of the group.

The *Group ID* attribute is required to identify a **Group** so that it can be referred to from other parts in the document. There is no predefined or required format for the *Group ID* as long as it uniquely identifies a group in the context of the current document. Examples include incremental integers or GUIDs.

#### Example

```
See examples in sections 5.1.4.1.1 and 5.1.4.1.2
```

#### 5.1.4.1.1 Product Tree – Product Groups – Group – Description

**prod:ProductTree / prod:ProductGroups / prod:Group / prod:Description**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Group |

The optional element `prod:Description` is a short, optional description of the group.

#### Example

*Example 32:*

```
<ProductGroups>
  <Group GroupID="CVRFGID-0001">
    <Description>The x64 versions of the operating system.</Description>
    <ProductID>CVRFPID-0001</ProductID>
    <ProductID>CVRFPID-0002</ProductID>
    <ProductID>CVRFPID-0003</ProductID>
    <ProductID>CVRFPID-0004</ProductID>
  </Group>
</ProductGroups>
```

### 5.1.4.1.2 Product Tree – Product Groups – Group – Product ID

**prod:ProductTree / prod:ProductGroups / prod:Group / prod:Description**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 2 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Group |

The element `prod:ProductID` defines a member of a group by referring to the unique *Product ID* attribute of a **Full Product Name** element.

#### Examples

*Example 33:*

If the two products “Microsoft Windows Vista Service Pack 1” and “Microsoft Windows Vista Service Pack 2” have been defined in the product tree as follows:

```
<FullProductName ProductID="CVRFPID-0001">
  Microsoft Windows Vista Service Pack 1
</FullProductName>
<FullProductName ProductID="CVRFPID-0002">
  Microsoft Windows Vista Service Pack 2
</FullProductName>
```

They can both be made a member of the same group with Group ID “GRP-0001”:

```
<ProductGroups>
  <Group GroupID="GRP-0001">
    <ProductID>CVRFPID-0001</ProductID>
    <ProductID>CVRFPID-0002</ProductID>
  </Group>
</ProductGroups>
```

Later in the document, both products can be referenced together using the Group ID:

```
<Remediations>
  <Remediation Type="Vendor Fix">
    <Description>Security Update for Windows Vista</Description>
    <GroupID>GRP-0001</GroupID>
  </Remediation>
</Remediations>
```

The ability to reference both products individually will also be maintained (and in some cases required):

```
<Remediations>
  <Remediation Type="Vendor Fix">
    <Description>Security Update for Windows Vista</Description>
    <ProductID>CVRFPID-0001</ProductID>
    <ProductID>CVRFPID-0002</ProductID>
  </Remediation>
</Remediations>
```

# 6 Vulnerability Schema Elements

Vulnerability information in CVRF is modeled as zero or more top-level Vulnerability element instances of `vuln:Vulnerability` (defined in the vulnerability schema file within the `vuln` namespace).

The following 14 second-level elements are and must appear in the order listed if given as elements of the top-level element Vulnerability:

|  |  |  |
|---|---|---|
| 5. | Title: | `vuln:Title` |
| 6. | ID: | `vuln:ID` |
| 7. | Notes: | `vuln:Notes` |
| 8. | Discovery Date: | `vuln:DiscoveryDate` |
| 9. | Release Date: | `vuln:ReleaseDate` |
| 10. | Involvements: | `vuln:Involvements` |
| 11. | CVE: | `vuln:CVE` |
| 12. | CWE: | `vuln:CWE` |
| 13. | Product Statuses: | `vuln:ProductStatuses` |
| 14. | Threats: | `vuln:Threats` |
| 15. | CVSS Score Sets: | `vuln:CVSSScoreSets` |
| 16. | Remediations: | `vuln:Remediations` |
| 17. | References: | `vuln:References` |
| 18. | Acknowledgements: | `vuln:Acknowledgements` |

The remaining sub sections will describe the above 15 first and second level elements together with their children and grandchildren, constraints on them as well as state recommendations and examples.

As a service to the reader otherwise commonly used acronyms stated above are expanded here and as understood in the context of CVRF (for more details please cf. the respective sub sections below):

- CVE: Common Vulnerabilities and Exposures
- CVSS: Common Vulnerability Scoring System (cf. [CVSS3] preferred, [CVSS2] deprecated)
- CWE: Common Weakness Enumeration

## 6.1 Vulnerability

**`vuln:Vulnerability`**

|  |  |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Root |
| **Children:** | Title, ID, Involvements, Notes, Discovery Date, Release Date, CVE, CWE, Threats, CVSS Score Sets, Remediation, Product Statuses, Acknowledgments, References |
| **Attribute:** | Ordinal |
| **Attribute Data Type:** | positiveInteger |
| **Attribute Required:** | yes |
| **Attribute Default Value:** | 1 |

The optional element `vuln:Vulnerability` is a container for the aggregation of all fields that are related to a single vulnerability in the document. There may be zero, one, or many vulnerabilities in a single CVRF document.

*Ordinal* is a locally significant value used to track vulnerabilities inside a CVRF document. It is provided to enable specific vulnerabilities to be referenced from elsewhere in the document (or even outside the namespace of a document provided that a unique **Document Title** and **Revision** information are provided). There should be one of these values for every **Vulnerability** container in a document, and it is recommended that *Ordinal* should be instantiated as a monotonically increasing counter, indexed from 1.

## Example

*Example 34:*

```
<Vulnerability Ordinal="1"
 xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/vuln">
 <!-- ... All children optional, so this is valid, albeit otherwise useless -->
</Vulnerability>
```

## Visual Overview

Map of **Vulnerability** element including the parent node (**Document**) in some valid configuration spanning multiple sub trees:

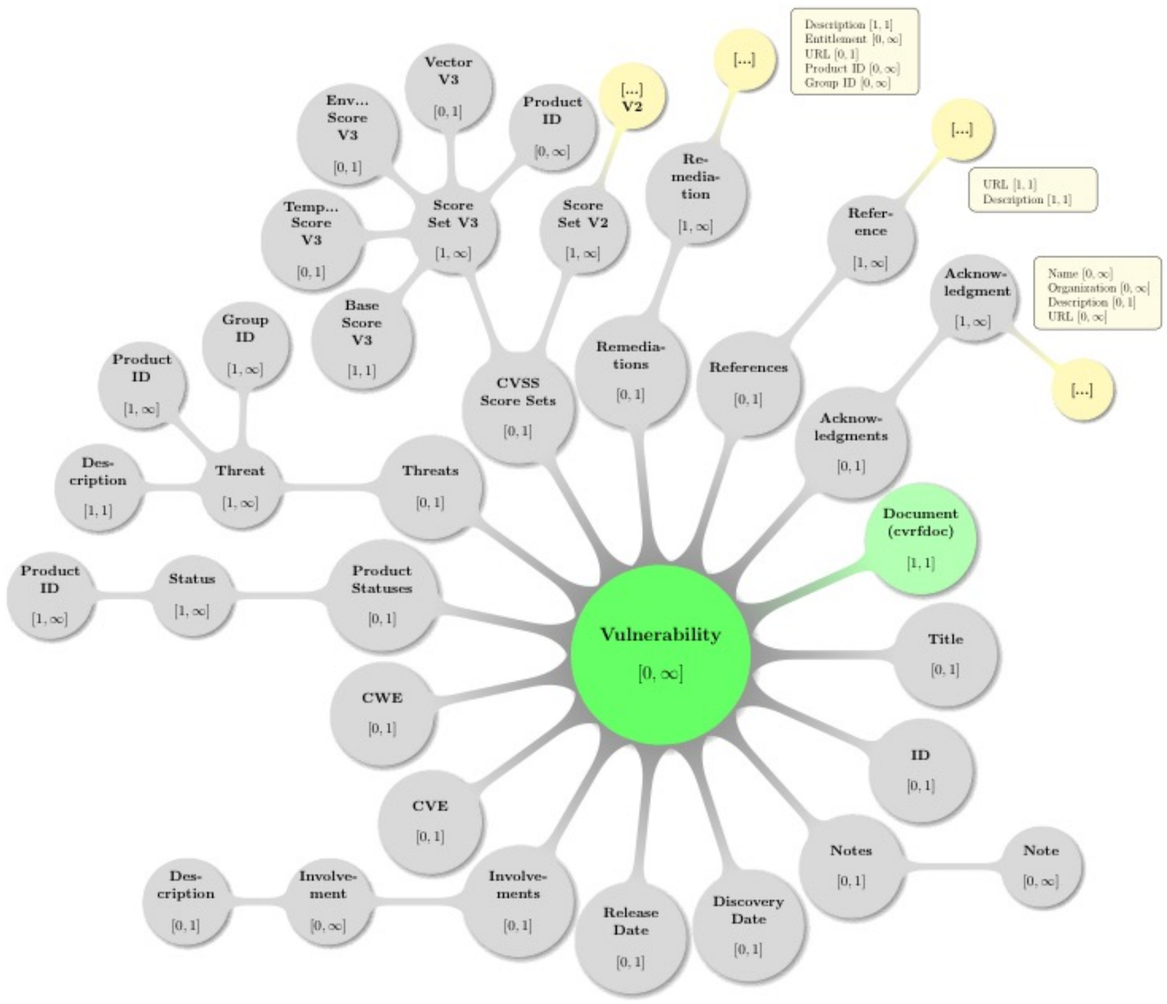

*Figure 4: Visual presentation of abstract but topologically valid Vulnerability instance.*

Some decent coloring has been applied to above graph as usual to balance visual hints with accessibility. Also, the mathematical closed interval notation has been used to annotate the minimum and maximum occurrences of elements, where the infinity symbol (∞) translates to the term unbounded in XML lingo.

The nodes carrying an ellipsis (…) here are to be read combined with the rounded edge rectangles, as the latter list the represented leaf elements that did not well fit into the picture.

## 6.2 Vulnerability – Title

**vuln:Vulnerability / vuln:Title**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |

The optional element `vuln:Title` gives the document producer the ability to apply a canonical name or title to the vulnerability. To avoid confusion, it is recommended that, if employed, this element commensurately match the nomenclature used by any numbering or cataloging systems references elsewhere, such as the **Document Title** or **CVE**.

### Examples

*Example 35:*

```
<Title>February 2011 TelePresence Vulnerability Bundle</Title>
```

## 6.3 Vulnerability – ID

**vuln:Vulnerability / vuln:ID**

| | |
|---|---|
| **Data Type:** | token |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Attribute:** | System Name |
| **Attribute Data Type:** | string |
| **Attribute Required:** | yes |

The optional element `vuln:ID` gives the document producer a place to publish a unique label or tracking ID for the vulnerability (if such information exists).

General examples may include an identifier from a vulnerability tracking system that is available to customers, such as a Cisco bug ID, an ID from a Bugzilla system, or an ID from a public vulnerability database such as the X-Force Database. The **ID** may be a vendor-specific value.

The **ID** should not be used for CVE tracking numbers (MITRE standard Common Vulnerabilities and Exposures). CVE numbers should be specified using the separate CVE element. Values are tokenized and can be alphanumeric.

The attribute *System Name* indicates the name of the vulnerability tracking or numbering system that this **ID** comes from. Every **ID** value should have exactly one *System Name*. It is helpful if document producers use unique and consistent system names.

### Example

*Example 36:*

```
<vuln:Vulnerability>
  <vuln:ID SystemName="Cisco Bug ID">CSCso66472</vuln:ID>
</vuln:Vulnerability>
```

## 6.4 Vulnerability – Notes

**`vuln:Vulnerabilty / vuln:Notes`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Children:** | Note |

The optional element `vuln:Notes` is a container that holds all vulnerability-level **Note** elements.

### Example

```
See example in section 4.5.1
```

### 6.4.1 Vulnerability – Notes – Note

**`vuln:Vulnerabilty / vuln:Notes / cvrf:Note`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Vulnerability |
| **Attribute:** | Type, Ordinal, Title, Audience |
| **Attribute Data Type:** | enumerated list, positiveInteger, string, string |
| **Attribute Required:** | yes, yes, no, no |

The element `vuln:Note` is a place to put all manner of text blobs related to the vulnerability. Text should be limited to talking about the impacts, vectors, or caveats of this node and should not contain details to other vulnerabilities in the document. It is, however, acceptable to refer to a vulnerability that is not in the document for the purposes of pointing out a regression.

Akin to the **Document Notes** element, the note should contain a compartmentalized textual discussion constrained by its *Type* attribute. *Type* can be one of the following:

- **General:** A general, high-level note (*Title* may have more information).
- **Details:** A low-level detailed discussion (*Title* may have more information).
- **Description:** A description of something (*Title* may have more information).
- **Summary**: A summary of something (*Title* may have more information).
- **FAQ**: A list of frequently asked questions.
- **Legal Disclaimer**: Any possible legal discussion, including constraints, surrounding the vulnerability.
- **Other**: Something that doesn't fit (*Title* should have more information).

*Title* and *Audience* are optional attributes to give human readers context around what they are about to read; *Title* should be a concise description of what is contained in the text, whereas *Audience* will indicate who is intended to read it.

*Ordinal* is a mandatory, locally significant value used to track notes inside a CVRF document at the vulnerability level. It is provided to uniquely identify a **Note**. There should be one of these values for every **Note** inside **Vulnerability/Notes** and it is recommended that *Ordinal* should be instantiated as a monotonically increasing counter, indexed from 1. Each *Ordinal* that tracks a **Note** inside **Vulnerability/Notes** is completely independent from an *Ordinal* tracking a **Note** inside **Document Notes**.

### Example

*Example 37:*

```
<vuln:Notes>
  <vuln:Note Type="General" Ordinal="1" Title="Details" Audience="All">
    These are some details about a vulnerability intended for all stakeholders.
  </vuln:Note>
</vuln:Notes>
```

## 6.5 Vulnerability – Discovery Date

**`vuln:Vulnerability / vuln:DiscoveryDate`**

| | |
|---|---|
| **Data Type:** | dateTime |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |

The optional element `vuln:DiscoveryDate` holds the date the vulnerability was originally discovered. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Universal Time Coordinated (UTC), also known as GMT or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

## 6.6 Vulnerability – Release Date

**`vuln:Vulnerability / vuln:ReleaseDate`**

| | |
|---|---|
| **Data Type:** | dateTime |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |

The optional element `vuln:ReleaseDate` holds the date the vulnerability was originally released into the wild. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Universal Time Coordinated (UTC), also known as GMT or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

## 6.7 Vulnerability – Involvements

**`vuln:Vulnerabilty / vuln:Involvements`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Children:** | Involvement |

The optional element `vuln:Involvements` is a container that holds one or more **Involvement** containers, which allow the document producers (or third party) to comment on their level of involvement in the vulnerability identification, scoping, and remediation process. Because there can be multiple Involvements containers, multiple parties can comment on their levels of involvement.

### Example

```
See example in section 6.7.1.1
```

### 6.7.1 Vulnerability – Involvements – Involvement

**`vuln:Vulnerabilty / vuln:Involvements / vuln:Involvement`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Involvements |
| **Children:** | Description |
| **Attribute:** | Party, Status |
| **Attribute Data Type:** | enumerated list, enumerated list |
| **Attribute Range:** | {Vendor, Discoverer, Coordinator, User, Other}, {Open, Disputed, In Progress, Completed, Contact Attempted, Not Contacted} |
| **Attribute Required:** | yes, yes |

The element `vuln:Involvement` is a container, that allows the document producers to comment on their level of Involvement (or engagement) in the vulnerability identification, scoping, and remediation process.

The attribute *Party* indicates the type of the producer issuing the status. It is identical to the **Document Publisher** attribute *Type*. Most of the time, both attributes will be the same because document producers will issue an **Involvement** status on their own behalf. However, if the document producer wants to issue a status on behalf of a third party and use a different type from that used in **Document Publisher**, that use is allowed by the schema. If this is the case, **Description** should contain additional context regarding what is going on.

The attribute *Status* indicates the level of involvement of *Party*. The child **Description** (below) is an optional element used to give context about the

involvement or engagement of the *Party*. The final two status states, “Contact Attempted” and “Not Contacted,” are intended for use by

document producers other than vendors (such as research or coordinating entities).

Status types include:

- **Open:** This is the default status. It doesn’t indicate anything about the vulnerability remediation effort other than the fact that the vendor has acknowledged awareness of the vulnerability report. The use of this status by a vendor indicates that future updates from the vendor about the vulnerability are to be expected.
- **Disputed:** This status indicates that the vendor disputes the vulnerability report in its entirety. Vendors should indicate this status when they believe that a vulnerability report regarding their product is completely inaccurate (that there is no real underlying security vulnerability) or that the technical issue being reported has no security implications.
- **In Progress:** This status indicates that some hotfixes, permanent fixes, mitigations, workarounds, or patches may have been made available by the vendor, but more information or fixes may be released in the future. The use of this status by a vendor indicates that future information from the vendor about the vulnerability is to be expected.

- **Completed:** The vendor asserts that investigation of the vulnerability is complete. No additional information, fixes, or documentation from the vendor about the vulnerability should be expected to be released.
- **Contact Attempted:** The document producer attempted to contact the affected vendor.
- **Not Contacted:** The document producer has not attempted to make contact with the affected vendor.

Each status is mutually exclusive—only one status is valid for a particular vulnerability at a particular time. As the vulnerability ages, a party's involvement could move from state to state. However, in many cases, a document producer may choose not to issue CVRF documents at each state, or simply omit this element altogether. It is recommended, however, that vendors that issue CVRF documents indicating an open or in-progress **Involvement** should eventually expect to issue a document as Disputed or Completed.

#### Example

`See example in section 6.7.1.1`

### 6.7.1.1 Vulnerability – Involvements – Involvement – Description

**`vuln:Vulnerabilty / vuln:... / vuln:Involvement / vuln:Description`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Involvement |

The optional element `vuln:Description` will contain a thorough human-readable discussion of the **Involvement**.

#### Examples

*Example 38:*

```
<Involvements>
  <Involvement Party="Vendor" Status="In Progress">
    <Description>
      Cisco acknowledges that the IronPort Email Security Appliances (ESA) and Cisco
      IronPort Security Management Appliances (SMA) contain a vulnerability that may
      allow a remote, unauthenticated attacker to execute arbitrary code with elevated
      privileges. A Mitigation is available.
    </Description>
  </Involvement>
</Involvements>
```

*Example 39:*

```
<Involvements>
  <Involvement Party="Researcher" Status="Contact Attempted">
    <Description>
      We emailed the vendor on February 14, 2012 when the vulnerability was first
      discovered by our team.
    </Description>
  </Involvement>
</Involvements>
```

## 6.8 Vulnerability – CVE

**`vuln:Vulnerability / vuln:CVE`**

| | |
|---|---|
| **Data Type:** | token |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |

The optional element `vuln:CVE` holds the MITRE standard Common Vulnerabilities and Exposures (CVE) tracking number for the vulnerability. CVE is a standard for vulnerability naming that provides improved tracking of vulnerabilities over time across different reporting sources. More information about CVE is available at http://cve.mitre.org/.

### Example

*Example 40:*

```
<CVE>CVE-2006-0010</CVE>
```

## 6.9 Vulnerability – CWE

**`vuln:Vulnerability / vuln:CWE`**

| | |
|---|---|
| **Data Type:** | token |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |

The optional element `vuln:CWE` contains the MITRE standard Common Weakness Enumeration (CWE). MITRE describes CWE in this way: “[CWE] is a formal list of software weakness types created to:

- Serve as a common language for describing software security weaknesses in architecture, design, or code.
- Serve as a standard measuring stick for software security tools targeting these weaknesses.
- Provide a common baseline standard for weakness identification, mitigation, and prevention efforts.”

More information about CWE is available at http://cwe.mitre.org/.

### Examples

*Example 41:*

```
<CWE ID="CWE-601">URL Redirection to Untrusted Site ('Open Redirect')</CWE>
```

*Example 42:*

```
<CWE ID="CWE-602">Client-Side Enforcement of Server-Side Security</CWE>
```

## 6.10 Vulnerability – Product Statuses

**`vuln:Vulnerabilty / vuln:ProductStatuses`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Children:** | Status |

The optional element `vuln:ProductStatuses` is a container that holds one or more **Status** containers, which will contain a subset of products chosen from **Product Tree** (see below). Each of the affected (and unaffected) products relating to the vulnerability will be referenced here, inside one or more **Status** containers.

Note there is a constraint in place to prevent a single product from being assigned two different (conflicting) **Status** elements within the scope of **Vulnerability**. Likewise, a **Status** child container cannot be tied to a **Product Group** due to the fact that a single product can be a member of more than one product group. Without this constraint, it would be possible to assign conflicting status information to one and the same product.

#### Example

```
See example in section 6.10.1.1
```

### 6.10.1 Vulnerability – Product Statuses – Status

**`vuln:Vulnerabilty / vuln:ProductStatuses / vuln:Status`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Product Statuses |
| **Children:** | Product ID |
| **Attribute:** | Type |
| **Attribute Data Type:** | enumerated list |
| **Attribute Required:** | yes |

The element `vuln:Status` contains one or more products as chosen from the **Product Tree**, and defines the status of this product in the mandatory *Status* attribute.

The *Type* attribute is an enumerated value that contains all the possible permutations of fixed, affected, and recommended versions of the products referenced inside the **Status** container. *Type* values include:

- **First Affected:** This is first version of the affected release known to be affected by the vulnerability.
- **Known Affected:** This version is known to be affected by the vulnerability.
- **Known Not Affected:** This version is known not to be affected by the vulnerability.
- **First Fixed:** This version contains the first fix for the vulnerability but may not be the recommended fixed version.
- **Fixed:** This version contains a fix for the vulnerability but may not be the recommended fixed version.
- **Recommended:** This version has a fix for the vulnerability and is the vendor-recommended version for fixing the vulnerability.
- **Last Affected:** This is the last version in a release train known to be affected by the vulnerability. Subsequently released versions would contain a fix for the vulnerability.

#### 6.10.1.1 Vulnerability – Product Statuses – Status – Product ID

**vuln:Vulnerabilty / vuln:ProductStatuses / vuln:Status / vuln:ProductID**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Status |

The element `vuln:ProductID` defines a product as having the status defined in the parent element's *Type* attribute. The reference is made using the unique *Product ID* attribute of a **Full Product Name** element that is defined in the **Product Tree**.

Note that a single **Product ID** may not be assigned more than one status type within the same **Vulnerability**.

### Example

*Example 43:*

The three products "Microsoft Windows Vista (RTM)", "Microsoft Windows Vista Service Pack 1", and "Microsoft Windows Vista Service Pack 2" have been defined in the product tree as follows:

```
<ProductTree>
  <FullProductName ProductID="CVRFPID-0000">
    Microsoft Windows Vista (RTM)
  </FullProductName>
  <FullProductName ProductID="CVRFPID-0001">
    Microsoft Windows Vista Service Pack 1
  </FullProductName>
  <FullProductName ProductID="CVRFPID-0002">
    Microsoft Windows Vista Service Pack 2
  </FullProductName>
</ProductTree>
```

If Windows Vista RTM and Service Pack 1 are known to be affected, and Service Pack 2 is known not to be affected, it can be documented as follows:

```
<Vulnerability Ordinal="1">
  <Product Statuses>
    <Status Type="KnownAffected">
      <ProductID>CVRFPID-0000</ProductID>
      <ProductID>CVRFPID-0001</ProductID>
    </Status>
    <Status Type="KnownNotAffected">
      <ProductID>CVRFPID-0002</ProductID>
    </Status>
  </Product Statuses>
</Vulnerability>
```

## 6.11 Vulnerability – Threats

**vuln:Vulnerabilty / vuln:Threats**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Children:** | Threat |

The optional element `vuln:Threats` is a container that holds one or more **Threat** containers, which contain information about a vulnerability that can change with time (so called "vulnerability kinetics").

### Example

```
See example in section 6.11.1.1
```

### 6.11.1 Vulnerability – Threats – Threat

**`vuln:Vulnerabilty / vuln:Threats / vuln:Threat`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Threats |
| **Children:** | Description, Product ID, Group ID |
| **Attribute:** | Type, Date |
| **Attribute Data Type:** | enumerated list, dateTime |
| **Attribute Required:** | yes, no |

The element `vuln:Threat` contains the vulnerability kinetic information. This information can change as the vulnerability ages and new information becomes available. A given **Threats** container can contain one or more **Threat**.

A **Threat** container can be tied to one or more specific products by referencing these products using either the **Product ID** or **Group ID** child elements. If the **Threat** is meant to be general or nonspecific for all products, the **Product ID** and **Group ID** child elements should be omitted.

The *Date* attribute is optional. All dateTime values in CVRF require a time, and we recommend the inclusion of a time zone as well (OASIS endorses the use of Universal Time Coordinated (UTC), also known as GMT or “Zulu time”). If a time zone is excluded, Zulu should be assumed.

The *Type* of Threat is required and can be one of the following:

**Impact:** Impact contains an assessment of the impact on the user or the target set if the vulnerability is successfully exploited. (A description of the **Target Set** *Type* follows.) If applicable, for consistency and simplicity, this section can be a textual summary of the three CVSS impact metrics. These metrics measure how a vulnerability detracts from the three core security properties of an information system: Confidentiality, Integrity, and Availability.

**Exploit Status:** Exploit Status contains a description of the degree to which an exploit for the vulnerability is known. This knowledge can range from information privately held among a very small group to an issue that has been described to the public at a major conference or is being widely exploited globally. For consistency and simplicity, this section can be a mirror image of the CVSS “Exploitability” metric. However, it can also contain a more contextual status, such as “Weaponized” or “Functioning Code.”

**Target Set:** Target Set contains a description of the currently known victim population in whatever terms are appropriate. Such terms may include: operating system platform, types of products, user segments, and geographic distribution.

### Example

```
See example in section 6.11.1.1
```

### 6.11.1.1 Vulnerability – Threats – Threat – Description

**`vuln:Vulnerabilty / vuln:Threats / vuln:Threat  / vuln:Description`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Threat |

The element `vuln:Description` contains a thorough human-readable discussion of the **Threat**.

#### Examples

*Example 44: Impact:*

```
<Threat Type="Impact">
  <Description>complete compromise of the integrity of affected machines </Description>
</Threat>
```

*Example 45: Exploit Status:*

```
<Threat Type="Exploit Status">
  <Description>none</Description>
  <Date>2011-11-26T00:00:00+00:00</Date>
  <ProductID>CVRFPID-0000</ProductID>
</Threat>
```

*Example 46: Exploit Status without Product ID:*

```
<Threat Type="Exploit Status">
  <Description>proof of concept</Description>
  </Date>2011-11-26T00:00:00+00:00</Date>
</Threat>
```

*Example 47: Target Set:*

```
<Threat Type="Target Set">
  <Description>Financial Institutions</Description>
</Threat>
```

*Example 48: Target Set variation of content:*

```
<Threat Type="Target Set">
  <Description>US Government Agencies</Description>
</Threat>
```

*Example 49: Target Set with another variation of content:*

```
<Threat Type="Target Set">
  <Description>All versions of BIND</Description>
</Threat>
```

### 6.11.1.2 Vulnerability – Threats – Threat – Product ID

**`vuln:Vulnerabilty / vuln:Threats / vuln:Threat  / vuln:ProductID`**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Threat |

If the **Threat** pertains to a specific product, an optional element `vuln:ProductID` can be added to reference that product. The reference is made using the unique *Product ID* attribute of a **Full Product**

**Name** element that is defined in the **Product Tree**. If a **Threat** applies to more than one Product, you can add multiple **Product ID** elements accordingly, or add the **Group ID** element (see below) instead.

> **Example**

```
See example in section 6.11.1.1
```

#### 6.11.1.3 Vulnerability – Threats – Threat – Group ID

> **`vuln:Vulnerabilty / vuln:Threats / vuln:Threat  / vuln:GroupID`**
>
> **Data Type:** token
> **Minimum Occurrences:** 0
> **Maximum Occurrences:** unbounded
> **Parent:** Threat

If the **Threat** pertains to several products that have been logically grouped into a **Product Group** optional element `vuln:GroupID` can be added to reference that group of products. The reference is made using the unique *Group ID* attribute of a **Group** element that is defined in the **Product Tree**. If a **Threat** applies to more than one group of products, you can add multiple **Group ID** elements accordingly.

## 6.12 Vulnerability – CVSS Score Sets

> **`vuln:Vulnerabilty / vuln:CVSSScoreSets`**
>
> **Data Type:** container
> **Minimum Occurrences:** 0
> **Maximum Occurrences:** 1
> **Parent:** Vulnerability
> **Children:** Score Set V2, Score Set V3

The optional element `vuln:CVSSScoreSets` is a container that holds container holds one or more of the **Score Set V2** or **Score Set V2** containers.

### 6.12.1 Vulnerability – CVSS Score Sets – Score Set V2

> **`vuln:Vulnerabilty / vuln:CVSSScoreSets / vuln:ScoreSetV2`**
>
> **Data Type:** container
> **Minimum Occurrences:** 0
> **Maximum Occurrences:** unbounded
> **Parent:** CVSS Score Sets
> **Children:** Base Score V2, Temporal Score V2, Environmental Score V2, Vector V2, Product ID

The optional element `vuln:ScoreSetV2` is a container that holds actual CVSS version 2 metrics [CVSS2]. The only required element of CVSS version 2 is the **Base Score**. If a value of the temporal or environmental score is set to “not defined,” either **Temporal Score** or **Environmental Score** can be omitted.

A **Score Set** container can be tied to one or more specific products by referencing these products using the **Product ID** child element. If the **Score Set** is meant to be applied for all products, the *Product ID* attribute should be omitted.

Note there is a constraint in place to prevent having a single product assigned to two different score sets within the scope of a **Vulnerability**. Likewise, a **Score Set** cannot be tied to a **Product Group** due to the fact that a single product can be a member of more than one product group. Without this constraint, it would be possible to assign conflicting base score information to one and the same product.

#### 6.12.1.1 Vulnerability – CVSS Score Sets – Score Set V2 – Base Score V2

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV2 / vuln:BaseScoreV2`**
>
> | | |
> |---|---|
> | **Data Type:** | float |
> | **Range:** | 0.0 – 10.0 |
> | **Minimum Occurrences:** | 1 |
> | **Maximum Occurrences:** | 1 |
> | **Parent:** | Score Set V2 |

The optional element `vuln:BaseScoreV2` contains the numeric value of the computed CVSS version 2 base score, which should be a float from 0 to 10.0.

#### 6.12.1.2 Vulnerability – CVSS Score Sets – Score Set V2 – Temporal Score V2

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV2 / vuln:TemporalScoreV2`**
>
> | | |
> |---|---|
> | **Data Type:** | float |
> | **Range:** | 0.0 – 10.0 |
> | **Minimum Occurrences:** | 0 |
> | **Maximum Occurrences:** | 1 |
> | **Parent:** | Score Set V2 |

The optional element `vuln:TemporalScoreV2` contains the numeric value of the computed CVSS version 2 temporal score, which should be a float from 0 to 10.0.

#### 6.12.1.3 Vulnerability – CVSS Score Sets – Score Set V2 – Environmental ScoreV2

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV2 / vuln:EnvironmentalScoreV2`**
>
> | | |
> |---|---|
> | **Data Type:** | float |
> | **Range:** | 0.0 – 10.0 |
> | **Minimum Occurrences:** | 0 |
> | **Maximum Occurrences:** | 1 |
> | **Parent:** | Score Set V2 |

The optional element `vuln:EnvironmentalScoreV2` contains the numeric value of the computed CVSS version 2 environmental score, which should be a float from 0 to 10.0. This metric is typically reserved for use by the end user and is specific to the environment in which the affected product is deployed.

#### 6.12.1.4 Vulnerability – CVSS Score Sets – Score Set V2 – Vector V2

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV2 / vuln:VectorV2`**
>
> | | |
> |---|---|
> | **Data Type:** | string |
> | **Range:** | 76 characters |
> | **Minimum Occurrences:** | 0 |
> | **Maximum Occurrences:** | 1 |
> | **Parent:** | Score Set V2 |

The optional element `vuln:VectorV2` contains the official notation that displays all the values used to compute the CVSS version 2 base, temporal, and environmental scores. This notation will follow the guidelines set forth in the CVSS v2 documentation at [CVSS2] (cf. section 2.4 “Base, Temporal, Environmental Vectors” there). Note the 76-character limitation.

#### Example

*Example 50:*

```
<VectorV2>AV:N/AC:L/Au:N/C:P/I:P/A:C/E:P/RL:O/RC:C/CDP:H/TD:M/CR:H/IR:H/AR:H<VectorV2>
```

### 6.12.1.5 Vulnerability – CVSS Score Sets – Score Set V2 – Product ID

**`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV2 / vuln:ProductID`**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Score Set V2 |

If the **Score Set** pertains to a specific product, a `vuln:ProductID` element can be added to reference that product. The reference is made using the unique *Product ID* attribute of a **Full Product Name** element that is defined in the **Product Tree**. If a **Score Set** applies to more than one product, you can add multiple **Product ID** elements accordingly.

Note that a single **Product ID** may not be assigned to more than one **Score Set** within the same **Vulnerability**.

## 6.12.2 Vulnerability – CVSS Score Sets – Score Set V3

**`vuln:Vulnerabilty / vuln:CVSSScoreSets / vuln:ScoreSetV3`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | CVSS Score Sets |
| **Children:** | Base Score, Temporal Score, Environmental Score, Vector, Product ID |

The optional element `vuln:ScoreSetV3` is a container that holds actual CVSS version 3 metrics [CVSS3]. The only required element of CVSS version 3 is the **Base Score**. If a value of the temporal or environmental score is set to “not defined,” either **Temporal Score** or **Environmental Score** can be omitted.

A **Score Set** container can be tied to one or more specific products by referencing these products using the **Product ID** child element. If the **Score Set** is meant to be applied for all products, the *Product ID* attribute should be omitted.

Note there is a constraint in place to prevent having a single product assigned to two different score sets within the scope of a **Vulnerability**. Likewise, a **Score Set** cannot be tied to a **Product Group** due to the fact that a single product can be a member of more than one product group. Without this constraint, it would be possible to assign conflicting base score information to one and the same product.

### 6.12.2.1 Vulnerability – CVSS Score Sets – Score Set V3 – Base Score V3

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV3 / vuln:BaseScoreV3`**
>
> **Data Type:** float
> **Range:** 0.0 – 10.0
> **Minimum Occurrences:** 1
> **Maximum Occurrences:** 1
> **Parent:** Score Set V3

The optional element `vuln:BaseScoreV3` contains the numeric value of the computed CVSS version 3 base score, which should be a float from 0 to 10.0.

### 6.12.2.2 Vulnerability – CVSS Score Sets – Score Set V3 – Temporal Score V3

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV3 / vuln:TemporalScoreV3`**
>
> **Data Type:** float
> **Range:** 0.0 – 10.0
> **Minimum Occurrences:** 0
> **Maximum Occurrences:** 1
> **Parent:** Score Set V3

The optional element `vuln:TemporalScoreV3` contains the numeric value of the computed CVSS version 3 temporal score, which should be a float from 0 to 10.0.

### 6.12.2.3 Vulnerability – CVSS Score Sets – Score Set V3 – Environmental ScoreV3

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV3 / vuln:EnvironmentalScoreV3`**
>
> **Data Type:** float
> **Range:** 0.0 – 10.0
> **Minimum Occurrences:** 0
> **Maximum Occurrences:** 1
> **Parent:** Score Set V3

The optional element `vuln:EnvironmentalScoreV3` contains the numeric value of the computed CVSS version 3 environmental score, which should be a float from 0 to 10.0. This metric is typically reserved for use by the end user and is specific to the environment in which the affected product is deployed.

### 6.12.2.4 Vulnerability – CVSS Score Sets – Score Set V3 – Vector V3

> **`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV3 / vuln:VectorV3`**
>
> **Data Type:** string
> **Range:** 76 characters
> **Minimum Occurrences:** 0
> **Maximum Occurrences:** 1
> **Parent:** Score Set V3

The optional element `vuln:VectorV3` contains the official notation that displays all the values used to compute the CVSS version 3 base, temporal, and environmental scores. This notation will follow the guidelines set forth in the CVSS v3 documentation at [CVSS3] (cf. section “Vector String” pp.17,18 there). Note the 76-character limitation.

### Example

*Example 51:*

```
<VectorV3>CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N<VectorV3>
```

*Example 52:*

```
<VectorV3>CVSS:3.0/S:U/AV:N/AC:L/PR:H/UI:N/C:L/I:L/A:N/E:F/RL:X<VectorV3>
```

#### 6.12.2.5 Vulnerability – CVSS Score Sets – Score Set V3 – Product ID

**`vuln:Vulnerabilty / vuln:... / vuln:ScoreSetV3 / vuln:ProductID`**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Score Set V3 |

If the **Score Set** pertains to a specific product, a `vuln:ProductID` element can be added to reference that product. The reference is made using the unique *Product ID* attribute of a **Full Product Name** element that is defined in the **Product Tree**. If a **Score Set** applies to more than one product, you can add multiple **Product ID** elements accordingly.

Note that a single **Product ID** may not be assigned to more than one **Score Set** within the same **Vulnerability**.

## 6.13 Vulnerability – Remediations

**`vuln:Vulnerabilty / vuln:Remediations`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Children:** | Remediation |

The optional element `vuln:Remediations` is a container that holds one or more **Remediation** containers, which will have details on how to remediate a vulnerability.

### 6.13.1 Vulnerability – Remediations – Remediation

**`vuln:Vulnerabilty / vuln:Remediations / vuln:Remediation`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Remediations |
| **Children:** | Description, Entitlement, URL, Product ID, Group ID |
| **Attribute:** | Type |
| **Attribute Data Type:** | enumerated list |
| **Attribute Range:** | {Workaround, Mitigation, Vendor Fix, None Available, Will Not Fix} |
| **Attribute Required:** | yes |

The optional element `vuln:Remediation` is a container that holds specific details on how to handle (and presumably, fix) a vulnerability. A given **Remediations** container can contain one or more **Remediation** containers.

A **Remediation** container can be tied to one or more specific products by referencing these products using either the **Product ID** or **Group ID** child elements. If the **Remediation** is meant to be general or nonspecific for all products, the **Product ID** and **Group ID** child elements should be omitted.

The *Type* attribute is required and can be one of the following:

- **Workaround:** Workaround contains information about a configuration or specific deployment scenario that can be used to avoid exposure to the vulnerability. There may be none, one, or more workarounds available. This is typically the "first line of defense" against a new vulnerability before a mitigation or vendor fix has been issued or even discovered.
- **Mitigation:** Mitigation contains information about a configuration or deployment scenario that helps to reduce the risk of the vulnerability but that does not resolve the vulnerability on the affected product. Mitigations may include using devices or access controls external to the affected product. Mitigations may or may not be issued by the original author of the affected product, and they may or may not be officially sanctioned by the document producer.
- **Vendor Fix:** Vendor Fix contains information about an official fix that is issued by the original author of the affected product. Unless otherwise noted, it is assumed that this fix fully resolves the vulnerability.
- **None Available:** Currently there is no fix available. Description should contain details about why there is no fix.
- **Will Not Fix:** There is no fix for the vulnerability and there never will be one. This is often the case when a product has been orphaned, end-of-lifed, or otherwise deprecated. Description should contain details about why there will be no fix issued.

Optionally, **Remediation** can contain information and constraints about how to obtain fixes via the **Entitlement** element.

#### 6.13.1.1 Vulnerability – Remediations – Remediation – Description

**`vuln:Vulnerabilty / vuln:... / vuln:Remediation / vuln:Description`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Remediation |

The element `vuln:Description` will contain a thorough human-readable discussion of the Remediation.

#### 6.13.1.2 Vulnerability – Remediations – Remediation – Entitlement

**`vuln:Vulnerabilty / vuln:... / vuln:Remediation / vuln:Entitlement`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Remediation |

The element `vuln:Entitlement` contains any possible vendor-defined constraints for obtaining fixed software or hardware that fully resolves the vulnerability. This element will often contain information about service contracts or service-level agreements that is directed toward customers of large vendors.

**Example**

*Example 53:*

```
<Entitlement>
```

```
  Cisco customers with service contracts that entitle them to regular software updates
  should obtain security fixes through their usual update channels, generally from the
  Cisco website. Cisco recommends contacting the TAC only with specific and imminent
  problems or questions.\r\nAs a special customer service, and to improve the overall
  security of the Internet, Cisco may offer customers free of charge software updates to
  address security problems. If Cisco has offered a free software update to address a
  specific issue, noncontract customers who are eligible for the update may obtain it by
  contacting the Cisco TAC using any of the means described in the Contact Summary
  section of this document. To verify their entitlement, individuals who contact the TAC
  should have available the URL of the Cisco document that is offering the
  upgrade.\r\nAll aspects of this process are subject to change without notice and on a
  case-by-case basis. No particular level of response is guaranteed for any specific
  issue or class of issues.
</Entitlement>
```

### 6.13.1.3 Vulnerability – Remediations – Remediation – URL

**`vuln:Vulnerabilty / vuln:... / vuln:Remediation / vuln:URL`**

| | |
|---|---|
| **Data Type:** | aniURI |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Remediation |

The optional element `vuln:URL` contains the URL to the Remediation.

#### Example

```
See example in section 6.13.1.4
```

### 6.13.1.4 Vulnerability – Remediations – Remediation – Product ID

**`vuln:Vulnerabilty / vuln:... / vuln:Remediation / vuln:ProductID`**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Remediation |

If the **Remediation** pertains to a specific product, a `vuln:ProductID` can be added to reference that product. The reference is made using the unique *Product ID* attribute of a **Full Product Name** element that is defined in the **Product Tree**. If a **Remediation** applies to more than one Product, you can add multiple **Product ID** elements accordingly, or add the **Group ID** element (see below) instead.

#### Example

*Example 54:*

```
<Remediation Type="Vendor Fix">
  <Description>
    this is an official fix for Test Product and here are the details...
  </Description>
  <URL>http://foo.foo/bar/</URL>
  <Product ID>CVRFPID-0000</Product ID>
</Remediation>
```

#### 6.13.1.5 Vulnerability – Remediations – Remediation – Group ID

**vuln:Vulnerabilty / vuln:... / vuln:Remediation / vuln:GroupID**

| | |
|---|---|
| **Data Type:** | token |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Remediation |

If the **Remediation** pertains to several products that have been logically grouped into a **Product Group**, a `vuln:GroupID` element can be added to reference that group of products. The reference is made using the unique *Group ID* attribute of a **Group** element that is defined in the **Product Tree**. If a **Remediation** applies to more than one group of products, you can add multiple **Group ID** elements accordingly.

## 6.14 Vulnerability – References

**vuln:Vulnerability / vuln:References**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Vulnerability |
| **Children:** | Reference |

The optional element `vuln:References` is a container that should include citations to any conferences, papers, advisories, and other resources that are specific to the vulnerability section and considered to be of value to the document consumer. For every **References** container, there must be at least one **Reference** element and each **Reference** element must contain one **URL** and one **Description.**

**Example**

```
See example in section 6.14.1.2
```

### 6.14.1 Vulnerability – References – Reference

**vuln:Vulnerability / vuln:References / vuln:Reference**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | References |
| **Children:** | URL, Description |
| **Attribute:** | Type |
| **Attribute Data Type:** | enumerated list |
| **Attribute Required:** | yes |
| **Attribute Default Value:** | External |

The element `vuln:Reference` contains a description of a related document specific to a vulnerability section of a CVRF document. This may include a plaintext or HTML version of the advisory or other related documentation, such as white papers or mitigation documentation.

The *Type* attribute denotes the type of the document reference relative to the CVRF document itself. The following types are available:

- **External**: The default value indicates the reference is external to the CVRF document.
- **Self**: This indicates the related document is actually a direct reference to itself

**Example**

```
See example in section 6.14.1.2
```

### 6.14.1.1 Vulnerability – References – Reference – URL

**`vuln:Vulnerability / vuln:References / vuln:Reference / vuln:URL`**

| | |
|---|---|
| **Data Type:** | anyURI |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Reference |

The element `vuln:URL` contains the fixed URL or location of the reference.

**Example**

```
See example in section 6.14.1.2
```

### 6.14.1.2 Vulnerability – References – Reference – Description

**`vuln:Vulnerability / vuln:References / vuln:Reference / vuln:Description`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Reference |

The element `vuln:Description` holds a descriptive title or name of the reference.

**Example**

*Example 55:*

```
<References>
  <Reference Type="External">
    <URL>http://foo.foo/bar/</URL>
    <Description xml:lang="fr">C'est un test de référence</Description>
  </Reference>
</References>
```

## 6.15 Vulnerability – Acknowledgements

**`vuln:Vulnerability / vuln:Acknowledgements`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Vulnerability |
| **Children:** | Acknowledgement |

The optional element `vuln:Acknowledgements` is a container that holds one or more **Acknowledgment** containers, which contain recognition of external parties. This **Acknowledgments** container is different from the one at the document level because it is specifically related to the vulnerability in question**.**

### 6.15.1 Vulnerability – Acknowledgements – Acknowledgement

**`vuln:Vulnerability / vuln:Acknowledgements / vuln:Acknowledgement`**

| | |
|---|---|
| **Data Type:** | container |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgements |
| **Children:** | Name, Organization, Description, URL |

The element `vuln:Acknowledgement` contains recognition of external parties who were instrumental in the discovery of, reporting of, and response to the vulnerability. This element indicates collaboration with the security community in a positive fashion and is an important part of a notice or advisory. Care should be taken to ensure that individuals would like to be acknowledged before they are included.

External parties who have worked with the document producer may be recognized for their work. This should be applied liberally; if someone reports an issue and then discloses it publicly, that party might still be credited.

If the original discoverer is not concerned with recognition, or the issue was discovered internally by the document producer, this field can be omitted.

An acknowledgment container may contain three different types of child elements: **Name**, **Organization**, and/or a **Description**. All are described below.

**Example**

```
See example in section 6.15.1.4
```

#### 6.15.1.1 Vulnerability – Acknowledgements – Acknowledgement – Name

**`vuln:Vulnerability / vuln:... / vuln:Acknowledgement / vuln:Name`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgement |

The element `vuln:Name` contains the name of the party being acknowledged.

**Example**

```
See example in section 6.15.1.4
```

#### 6.15.1.2 Vulnerability – Acknowledgements – Acknowledgement – Organization

**`vuln:Vulnerability / vuln:... / vuln:Acknowledgement / vuln:Organization`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 1 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgement |

The element `vuln:Organization` contains the organization of the party or, if the Name is omitted, the organization itself that is being acknowledged.

**Example**

```
See example in section 6.15.1.4
```

### 6.15.1.3 Vulnerability – Acknowledgements – Acknowledgement – Description

**`vuln:Vulnerability / vuln:... / vuln:Acknowledgement / vuln:Description`**

| | |
|---|---|
| **Data Type:** | string |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | 1 |
| **Parent:** | Acknowledgement |

The element `vuln:Description` can contain any contextual details the document producers wish to make known about the acknowledgment or acknowledged parties.

If attributing to multiple organizations, each contributor should be grouped with that **Organization** within a single **Acknowledgment** container. An **Organization**-specific acknowledgment may be added within each **Acknowledgment** container by the **Description** element. If an overall general or aggregate acknowledgment is to be added, an **Acknowledgment** container that contains a single **Description** element may be used.

**Example**

```
See example in section 6.15.1.4
```

### 6.15.1.4 Vulnerability – Acknowledgements – Acknowledgement – URL

**`vuln:Vulnerability / vuln:... / vuln:Acknowledgement / vuln:URL`**

| | |
|---|---|
| **Data Type:** | anyURI |
| **Range:** | unrestricted |
| **Minimum Occurrences:** | 0 |
| **Maximum Occurrences:** | unbounded |
| **Parent:** | Acknowledgement |

The element `vuln:URL` contains the URL or location of the reference to be acknowledged.

**Example**

*Example 56:*

```
<Acknowledgments>
  <Acknowledgment>
    <Name>[Name 1]</Name>
    <Name>[Name 2]</Name>
    <Organization>[OrgName]</Organization>
    <URL>http://foo.foo/bar/</URL>
  </Acknowledgment>
  <Acknowledgment>
    <Name>[Name 3]</Name>
    <Organization>[OrgName]</Organization>
    <Description>
```

```
      Vendor X would like to thank [Name 3] from [OrgName] for reporting this issue.
    </Description>
    <URL>http://foo.foo/bar/</URL>
  </Acknowledgment>
  <Acknowledgment>
    <Description>
      Vendor X would like to thank the following researchers for their contributions to
      making this project more secure: [Name 1], [Name 2], [Name 3]
    </Description>
    <URL>http://foo.foo/bar/</URL>
  </Acknowledgment>
</Acknowledgments>
```

# 7 Conformance

## 7.1 Conformance as a CVRF version 1.2 document

A document instance conforms to this specification as a CVRF version 1.2 document if it meets the following three conditions:

1. Is well-formed XML.
2. Consists of a single `cvrf:cvrfdoc` element instance as defined in the namespace `http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf`.
3. Is valid XML.

# Appendix A. Acknowledgments

The following individuals were members of the OASIS CSAF Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

Adam Montville, CIS
Allan Thomson, LookingGlass
Anthony Berglas, Cryptsoft Pty Ltd.
Art Manion, Carnegie Mellon University
Aukjan van Belkum, EclecticIQ
Bernd Grobauer, Siemens AG
Beth Pumo, Kaiser Permanente
Bret Jordan, Symantec Corp.
Bruce Rich, Cryptsoft Pty Ltd.
Chet Ensign, OASIS
Chok Poh, Oracle
Chris Rouland, Individual
David Waltermire, NIST
Denny Page, TIBCO Software Inc.
Doron Shiloach, IBM
Duncan Sparrell, sFractal Consulting LLC
Eric Johnson, TIBCO Software Inc.
Feng Cao, Oracle
Greg Reaume, TELUS
Greg Scott, Cryptsoft Pty Ltd.
Harold Booth, NIST
Jamison Day, LookingGlass
Jared Semrau, "FireEye, Inc."
Jason Keirstead, IBM
Jason Masters, TELUS
Jerome Athias, Individual
Jessica Fitzgerald-McKay, National Security Agency
Jonathan Bitle, Kaiser Permanente
Justin Corlett, Cryptsoft Pty Ltd.
Karen Scarfone, Individual
Kazuo Noguchi, "Hitachi, Ltd."
Kent Landfield, Intel Corporation
Lothar Braun, Siemens AG
Louis Ronnau, Cisco Systems
Mark Davidson, NC4
Mark-David McLaughlin, Cisco Systems
Masato Terada, "Hitachi, Ltd."
Nicole Gong, Mitre Corporation
Omar Santos, Cisco Systems
Patrick Maroney, Wapack Labs LLC
Paul Patrick, "FireEye, Inc."
Peter Allor, IBM
Phillip Boles, "FireEye, Inc."
Ravi Balupari, Netskope
Rich Reybok, ServiceNow
Richard Struse, DHS Office of Cybersecurity and Communications (CS&C)
Ritwik Ghoshal, Oracle
Robert Coderre, VeriSign
Robin Cover, OASIS
Rupert Wimmer, Siemens AG
Sanjiv Kalkar, Individual
Sarah Kelley, CIS
Sean Barnum, Mitre Corporation
Stefan Hagen, Individual
Ted Bedwell, Cisco Systems
Thomas Schreck, Siemens AG
Tim Hudson, Cryptsoft Pty Ltd.
Tony Cox, Cryptsoft Pty Ltd.
Trey Darley, "Kingfisher Operations, sprl"
Troy Fridley, Cisco Systems
Vincent Danen, Red Hat
Zach Turk, Microsoft

# Appendix B. Examples

Some real-world examples of CVRF version 1.2 advisories in XML format are given in this appendix and in the hope, they are useful (major edits on text content to emphasize structure).

## B.1 Sample Security Advisory A

Security advisory from the year 2017:

```
<?xml version="1.0" encoding="UTF-8"?>
<cvrf:cvrfdoc
  xmlns:cpe="http://cpe.mitre.org/language/2.0"
  xmlns:cvrf="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf"
  xmlns:cvrf-common="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/common"
  xmlns:cvssv2="http://scap.nist.gov/schema/cvss-v2/1.0"
  xmlns:cvssv3="https://www.first.org/cvss/cvss-v3.0.xsd"
  xmlns:dc="http://purl.org/dc/elements/1.2/"
  xmlns:ns0="http://purl.org/dc/elements/1.1/"
  xmlns:prod="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/prod"
  xmlns:scap-core="http://scap.nist.gov/schema/scap-core/1.0"
  xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  xmlns:vuln="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/vuln"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf
  >
  <!-- Document wide context info -->
  <cvrf:DocumentTitle>AppY Stream Control Transmission Protocol</cvrf:DocumentTitle>
  <cvrf:DocumentType>Security Advisory</cvrf:DocumentType>
  <cvrf:DocumentPublisher Type="Vendor">
      <cvrf:ContactDetails>Emergency Support: ...</cvrf:ContactDetails>
      <cvrf:IssuingAuthority>... Team (PSIRT)....</cvrf:IssuingAuthority>
  </cvrf:DocumentPublisher>
  <cvrf:DocumentTracking>
    <cvrf:Identification>
      <cvrf:ID>vendorix-sa-20170301-abc</cvrf:ID>
    </cvrf:Identification>
    <cvrf:Status>Final</cvrf:Status>
    <cvrf:Version>1.0</cvrf:Version>
    <cvrf:RevisionHistory>
      <cvrf:Revision>
        <cvrf:Number>1.0</cvrf:Number>
        <cvrf:Date>2017-03-01T14:58:48</cvrf:Date>
        <cvrf:Description>Initial public release.</cvrf:Description>
      </cvrf:Revision>
    </cvrf:RevisionHistory>
    <cvrf:InitialReleaseDate>2017-03-01T16:00:00</cvrf:InitialReleaseDate>
    <cvrf:CurrentReleaseDate>2017-03-01T14:58:48</cvrf:CurrentReleaseDate>
    <cvrf:Generator>
      <cvrf:Engine>TVCE</cvrf:Engine>
    </cvrf:Generator>
  </cvrf:DocumentTracking>
  <cvrf:DocumentNotes>
    <cvrf:Note Title="Summary" Type="General" Ordinal="1">A vulnerability...</cvrf:Note>
    <cvrf:Note Title="CVSS 3.0 Notice" Type="Other" Ordinal="2">... </cvrf:Note>
  </cvrf:DocumentNotes>
  <cvrf:DocumentReferences>
    <cvrf:Reference Type="Self">
      <cvrf:URL>https://example.com/sec/vendorix-sa-20170301-abc</cvrf:URL>
      <cvrf:Description>Vendorix Foo AppY...</cvrf:Description>
    </cvrf:Reference>
  </cvrf:DocumentReferences>
  <!-- Product tree section -->
  <prod:ProductTree>
    <prod:Branch Name="Vendorix" Type="Vendor">
      <prod:Branch Name="... Appliances" Type="Product Name">
        <prod:Branch Name="1.0" Type="Product Version">
          <prod:Branch Name=".0" Type="Service Pack">
                <prod:FullProductName ProductID="CVRFPID-223152">...
                  AppY 1.0.0</prod:FullProductName>
          </prod:Branch>
          <prod:Branch Name="(2)" Type="Service Pack">
                <prod:FullProductName ProductID="CVRFPID-223153">...
                  AppY 1.0(2)</prod:FullProductName>
          </prod:Branch>
        </prod:Branch>
        <prod:Branch Name="1.1" Type="Product Version">
          <prod:Branch Name=".0" Type="Service Pack">
                <prod:FullProductName ProductID="CVRFPID-223155">...
                  AppY 1.1.0</prod:FullProductName>
          </prod:Branch>
              <prod:Branch Name="(1)" Type="Service Pack">
                <prod:FullProductName ProductID="CVRFPID-223156">...
                  AppY 1.1(1)</prod:FullProductName>
              </prod:Branch>
        </prod:Branch>
      </prod:Branch>
    </prod:Branch>
  </prod:ProductTree>
  <!-- Vulnerability section -->
  <vuln:Vulnerability Ordinal="1">
    <vuln:Title>... Transmission Protocol  ...</vuln:Title>
    <vuln:ID SystemName="Vendorix Bug ID">VDXvc83320</vuln:ID>
    <vuln:Notes>
      <vuln:Note Title="Summary" Type="Summary" Ordinal="1">A vuln ...</vuln:Note>
      <vuln:Note Title="Vendorix Bug IDs" Type="Other" Ordinal="3">
        VDXvc83320</vuln:Note>
    </vuln:Notes>
    <vuln:CVE>CVE-2017-3826</vuln:CVE>
    <vuln:ProductStatuses>
      <vuln:Status Type="Known Affected">
        <vuln:ProductID>CVRFPID-223152</vuln:ProductID>
        <vuln:ProductID>CVRFPID-223153</vuln:ProductID>
        <vuln:ProductID>CVRFPID-223155</vuln:ProductID>
        <vuln:ProductID>CVRFPID-223156</vuln:ProductID>
      </vuln:Status>
    </vuln:ProductStatuses>
    <vuln:CVSSScoreSets>
      <vuln:ScoreSetV3>
        <vuln:BaseScoreV3>7.5</vuln:BaseScoreV3>
        <vuln:VectorV3>CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H</vuln:VectorV3>
         </vuln:ScoreSetV3>
      </vuln:CVSSScoreSets>
    <vuln:Remediations>
      <vuln:Remediation Type="Workaround">
        <vuln:Description>There are no workarounds that ...</vuln:Description>
          </vuln:Remediation>
        </vuln:Remediations>
    <vuln:References>
      <vuln:Reference Type="Self">
        <vuln:URL>https://example.com/sec/vendorix-sa-20170301-abc</vuln:URL>
        <vuln:Description>... AppY Stream ...</vuln:Description>
      </vuln:Reference>
    </vuln:References>
  </vuln:Vulnerability>
  <!-- No more elements to follow -->
</cvrf:cvrfdoc>
```

## B.2 Sample Security Advisory B

Another security advisory from the year 2017:

```
TODO
```

## B.3 Sample Security Advisory C

Minimal valid CSAF CVRF version 1.2 document (not useful as neither Product Tree nor Vulnerability noted, but nevertheless valid):

```
TODO
```

## B.4 Sample Security Advisory D

Minimal viable product like fictitious sample valid CSAF CVRF version 1.2 document:

```
TODO
```

Note: The vendor is assumed to be named ACME Inc., only hosted on subdomains of acme.example.com. A product foo is declared to be available on platform bar and baz alike. Some possible combinations of vulnerability and mitigation states are realized.

# Appendix C. Revision History

| Revision | Date | Editor | Changes Made |
|---|---|---|---|
| Working Draft 01 | 2017-03-10 | Stefan Hagen | Combined and migrated CVRF version 1.1 contribution to OASIS CSAF CVRF version 1.2 |