CSAF Common Vulnerability Reporting Framework (CVRF) Version 1.2

Working Draft 01

24 May 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.*, ICASI, ICASI CVRF Working Group, Mike Schiffman, Editor, May 2012, 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 CSAF CVRF language which supports creation, update, and interoperable exchange of security advisories as structured information on products, vulnerabilities and the status of impact and remediation among interested parties.

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

1.1 Organization of CSAF CVRF

The specification is split into seven chapters.

1.2 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].

1.2.1 Terms and Definitions

For the purposes of this document, the following applies:

Advisory	— as in ISO/IEC 29147 [ISO29147]
Coordinator	— as in ISO/IEC 29147 [ISO29147]
Finder	— as in ISO/IEC 29147 [ISO29147]
Online Services	— as in ISO/IEC 29147 [ISO29147]
Product	— as in ISO/IEC 29147 [ISO29147]
Remediation	— as in ISO/IEC 29147 [ISO29147]
Service	— as in ISO/IEC 29147 [ISO29147]
Vendor	— as in ISO/IEC 29147 [ISO29147]
Vulnerability	— as in ISO/IEC 29147 [ISO29147]
Date Time Value	— as defined in section 2.2.1 Date and Time
Positive Integer	— integer number > 0 (`xs:positiveInteger`) cf. [XML-Schema-2]
Cardinality Range	— minimal and maximal occurrence noted as mathematical closed interval with the special symbol (∞) for infinity to indicate no upper limit.
Enumeration Set	— a finite set of domain values (case-sensitive string literals) implemented via enumerations on (`xs:token`) cf. [XML-Schema-2]

1.2.2 Abbreviated Terms

CCE	— Common Configuration Enumeration
CSAF	— Common Security Advisory Framework
CPE	— Common Platform Enumeration [CPE23-N]
CSIRT	— Computer Security Incident Response Team
CVE	— Common Vulnerabilities and Exposures [CVE]
CVRF	— Common Vulnerability Reporting Framework
CVSS	— Common Vulnerability Scoring System [CVSS]
CWE	— Common Weakness Enumeration [CWE]
ICASI	— Internet Consortium for Advancement of Security on the Internet, http://www.icasi.org/
ID	— Identifier

OASIS	— Organization for the Advancement of Structured Information Standards, https://www.oasis-open.org/
PSIRT	— Product Security Incident Response Team
URL	— Uniform Resource Locator
UTC	— Coordinated Universal Time (Acronym: Universal Time Coordinated) [ISO8601]

1.3 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/.

1.4 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.

[CVE] Common Vulnerability and Exposures (CVE) – The Standard for Information Security Vulnerability Names, MITRE, 1999, https://cve.mitre.org/about/.

[CVE-NF] Common Vulnerability and Exposures (CVE) – The Standard for Information Security Vulnerability Names - CVE ID Syntax Change, MITRE, January 01, 2014, https://cve.mitre.org/cve/identifiers/syntaxchange.html.

[CVRF-1-1] The Common Vulnerability Reporting Framework (CVRF) Version 1.1, M. Schiffman, Editor, May 2012, Internet Consortium for Advancement of Security on the Internet (ICASI), http://www.icasi.org/the-common-vulnerability-reporting-framework-cvrf-v1-1/.

[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.
[CWE]	Common Weakness Enumeration (CWE) – A Community-Developed List of Software Weakness Types, MITRE, 2005, http://cwe.mitre.org/about/.
[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/.
[ISO8601]	Data elements and interchange formats — Information interchange — Representation of dates and times, International Standard, ISO 8601:2004(E), December 1, 2004, https://www.iso.org/standard/40874.html.
[ISO29147]	Information technology — Security techniques — Vulnerability disclosure, International Standard, ISO 29147:2014(E), February 15, 2014, https://www.iso.org/standard/45170.html.
[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.

1.5 Typographical Conventions

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

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

Text following the special symbol («) – an opening Guillemet (or French quotation mark) – within this specification identifies conformance statements. Every conformance statement is separated from the following text with the special end symbol (») – a closing Guillemet, and has been assigned a reference that follows that end symbol in the format [CSAF-section#-local#].

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 e.g. like:

Non-normative Comment:

This is a pure informative comment that may be present, because the information conveyed is deemed useful advice or common pitfalls learned from implementer or operator experience and often given including the rationale.

2 Design Considerations

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

Non-normative comment:

> CVRF version 1.1 has been contributed to OASIS for future enhancement by its Common Security Advisory Framework (CSAF) Technical Committee (TC).

The term Security Advisory as used in this document describes any notification of security issues in products of and by providers. Anyone providing a product is considered in this document as a vendor, i.e. 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.

A security issue is not necessarily constraint to a problem statement, the focus of the term is on the security aspect impacting (or **not** impacting) specific product-platform-version combinations. Information on **presence** or **absence** of work-arounds is also considered part of the security issue.

This document is the definitive reference for the language elements of CSAF CVRF 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 CSAF 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 CSAF CVRF 1.2 is **not** backward compatible with CVRF 1.1 published by ICASI [CVRF-1-1] 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 CSAF 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`"): » [CSAF-2-1]

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 information 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 information 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** information has been placed in the sections named accordingly.

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 E Complete Examples.

2.1 Construction Principles

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 utilized by the providers of 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.

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

1. The frame, aggregation, and reference information 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 information to the matching product.

A dictionary like presentation of all defined schema elements 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 CSAF CVRF documents. It is out of scope for this specification.

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`

2.2 Domain Models

2.2.1 Date and Time Model

« All date time values inside a CSAF CVRF document MUST adhere to the ISO 8601 [ISO8601] basic or extended Format (as given there in section 4.3.2 "Complete representations" and with the addition of decimal fractions for seconds, similar to ibid. section 4.2.2.4 "Representations with decimal fraction" but with the full stop (.) being the preferred separator for CSAF CVRF). » [CSAF-2.2.1-1].

Many CSAF CVRF documents are considered to be shared messages with distributed incremental update and forwarding cycles. Coordinated Universal Time (UTC) is the best fit time system for world-wide exchanged and used date time information.

« To ensure maximal interoperability any date time literal having an empty zone designator MUST be treated as having UTC offset 0 or equivalently[2] as if the zone designator would have been the UTC designator (Z). » [CSAF-2.2.1-2]

The following CSAF CVRF date time literals expressed in the language of the ISO8601 abstract representations for digits of year(Y), month(M), day(D), hour(h), minute(m), seconds(s), and the special marker (T) are suggested for maximal interoperability in exchange:

Without fractional second digits (also no "full stop" separator):

```
YYYYMMDDThhmmssZ
YYYYMMDDThhmmss+hhmm
YYYY-MM-DDThh:mm:ssZ
YYYY-MM-DDThh:mm:ss+hh:mm
```

Including fractional second digits:

```
YYYYMMDDThhmmss.sZ
YYYYMMDDThhmmss.s+hhmm
YYYY-MM-DDThh:mm:ss.sZ
YYYY-MM-DDThh:mm:ss.s+hh:mm
```

« The T separator literal MUST be kept, as leaving it out it is not expected to safe significant space but instead challenge interoperability. » [CSAF-2.2.1-3]

Note: Time zone calculations are not considered to be in scope for this specification.

Example 2: Basic format results in April 30, 1985 at time 23:15:30.0 UTC (due to Z as zone designator for UTC)

```
19850430T231530.0Z
```

Example 3: Basic format results in April 30, 1985 at time 23:15:30.0 UTC (due to +0400 offset i.e. 4 hours' positive difference between the time scale of local time and UTC)

```
19850501T031530.0+0400
```

Example 4: Extended format results in April 30, 1985 at time 23:15:30.0 UTC (due to Z as zone designator for UTC)

```
1985-04-30T23:15:30.0Z
```

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

[2] Here we deviate from ISO8601 where in section 4.3.2 "Complete representations" at the start of the first paragraph after the initial list that: "The zone designator is empty if use is made of local time ..." is stated.

Example 5: Extended format for April 30, 1985 at time 23:15:30.0 UTC (due to +01:00 offset i.e. 1 hour' positive difference between the time scale of local time and UTC)

```
1985-05-01T00:15:30.0+01:00
```

2.2.2 Note Type Model

In the scope of CSAF CVRF, a note is considered an element, that holds all manner of text blobs related to either a surrounding part of or the entire document.

A note is anything between a concise summary of the target area and a more compartmentalized and area-specific textual discussion.

The note SHOULD contain a compartmentalized textual discussion constrained by its Type attribute.

« Any value for the **Type** attribute of any CSAF CVRF Note element (regardless of its parent element) MUST be one of following seven categories:

```
Description
Details
FAQ
General
Legal Disclaimer
Other
Summary
```

» [CSAF-2.2.2-1]

A Note may also be annotated with a `Title` and an `Audience` attribute. These optional attributes are intended 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 of the note, whereas Audience will indicate who is intended to read it.

Non-normative comment:

In the following sub sections the terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.2.1 Note Type Model — Description

The value `Description` of `Note Type` indicates the note is a description of something. The optional sibling attribute `Title` MAY have more information in this case.

2.2.2.2 Note Type Model — Details

The value `Details` of `Note Type` indicates the note is a low-level detailed discussion. The optional sibling attribute `Title` MAY have more information in this case.

2.2.2.3 Note Type Model — FAQ

The value `FAQ` of `Note Type` indicates the note is a list of frequently asked questions.

2.2.2.4 Note Type Model — General

The value `General` of `Note Type` indicates the note is a general, high-level note. The optional sibling attribute `Title` MAY have more information in this case.

2.2.2.5 Note Type Model — Legal Disclaimer

The value `Legal Disclaimer` of `Note Type` indicates the note represents any possible legal discussion, including constraints, surrounding the document.

2.2.2.6 Note Type Model — Other

The value `Other` of `Note Type` indicates the note is something that doesn't fit the other categories. The optional sibling attribute `Title` SHOULD have more information to indicate clearly what kind of note to expect in this case.

2.2.2.7 Note Type Model — Summary

The value `Summary` of `Note Type` indicates the note is a summary of something. The optional sibling attribute `Title` MAY have more information in this case.

2.2.3 Product Branch Type Model

To model **Product Branches** the **CSAF CVRF** chosen modelling offers `Type-Name` annotation pairs to describe the characteristics of the labeled **Branch,** where the `Type` part is restricted to a fixed vocabulary (an enumeration) and the `Name` will contain the canonical descriptor or "friendly name" of the **Branch**.

Thus, the `Type` attribute will describe the category of the **Branch** in question.

« The following eleven categories MUST be used as values for **Branch Type** fields in CSAF CVRF – case-sensitive and including the space between the words if multi word term:

```
Architecture
Host Name
Language
Legacy
Patch Level
Product Family
Product Name
Product Version
Service Pack
Specification
Vendor
```

» [CSAF-2.2.3-1]

Non-normative comment:

Short descriptions of these eleven values are given in the following subsections.

2.2.3.1 Product Branch Type Model — Architecture

The value `Architecture` of `Branch Type` indicates the architecture for which the product is intended.

2.2.3.2 Product Branch Type Model — Host Name

The value `Host Name` of `Branch Type` indicates the host name of a system/service.

2.2.3.3 Product Branch Type Model — Language

The value `Language` of `Branch Type` indicates the language of the product.

2.2.3.4 Product Branch Type Model — Legacy

The value `Legacy` of `Branch Type` indicates a nonspecific legacy entry.

2.2.3.5 Product Branch Type Model — Patch Level

The value `Patch Level` of `Branch Type` indicates the patch level of the product.

2.2.3.6 Product Branch Type Model — Product Family

The value `Product Family` of `Branch Type` indicates the product family that the product falls into.

2.2.3.7 Product Branch Type Model — Product Name

The value `Produzct Name` of `Branch Type` indicates the name of the product.

2.2.3.8 Product Branch Type Model — Product Version

The value `Product Version` of `Branch Type` indicates the product version, can be numeric or some other descriptor.

2.2.3.9 Product Branch Type Model — Service Pack

The value `Service Pack` of `Branch Type` indicates the service pack of the product.

2.2.3.10 Product Branch Type Model — Specification

The value `Specification` of `Branch Type` indicates the specification such as a standard, best common practice, etc.

2.2.3.11 Product Branch Type Model — Vendor

The value `Vendor` of `Branch Type` indicates the name of the vendor or manufacturer that makes the product.

2.2.4 Product Relationship Type Model

The concept of a **Product Relationship Type** allows and guides security advisories to express situations with higher order dependencies of vulnerabilities on product combinations in CSAF CVRF documents.

Relationships live at the root of a **Product Tree**, and they have three mandatory attributes: `ProductReference` and `RelatesToProductReference` 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.

« The following five categories MUST be used as values for **Product Relationship** `Type` attributes in CSAF CVRF – exactly as written and including the space between words if multi word terms:

```
    Default Component Of
    External Component Of
    Installed On
    Installed With
    Optional Component Of
```

» [CSAF-2.2.4-1]

Non-normative comment:

Situations with higher order dependencies of vulnerabilities on product combinations arise when a product is vulnerable only when installed together with another, or to describe operating system components. As such 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.

Once a **Relationship** element has been created, it is expected to be completed by adding one **Full Product Name** element to it, typically using a combination of the two related product names as a value.

Short descriptions of these eleven values are given in the following subsections, with examples based on the assumption, that two **Product IDs** are given as `CVRFPID-0001` and `CVRFPID-0002`.

2.2.4.1 Product Relationship Type Model — Default Component Of

The value `Default Component Of` stored in `Relationship Type` indicates that the entity labeled with one Product ID (e.g. `CVRFPID-0001`) is a default component of an entity with another Product ID (e.g. `CVRFPID-0002`). These Product ID's SHOULD NOT be identical to provide minimal redundancy.

2.2.4.2 Product Relationship Type Model — External Component Of

The value `External Component Of` stored in `Relationship Type` indicates that the entity labeled with one Product ID (e.g. `CVRFPID-0001`) is an external component of an entity with another Product ID (e.g. `CVRFPID-0002`). These Product ID's SHOULD NOT be identical to provide minimal redundancy.

2.2.4.3 Product Relationship Type Model — Installed On

The value `Installed On` stored in `Relationship Type` indicates that the entity labeled with one Product ID (e.g. `CVRFPID-0001`) is installed on a platform entity with another Product ID (e.g. `CVRFPID-0002`). These Product ID's SHOULD NOT be identical to provide minimal redundancy.

2.2.4.4 Product Relationship Type Model — Installed With

The value `Installed With` stored in `Relationship Type` indicates that the entity labeled with one Product ID (e.g. `CVRFPID-0001`) is installed alongside an entity with another Product ID (e.g. `CVRFPID-0002`). These Product ID's SHOULD NOT be identical to provide minimal redundancy.

2.2.4.5 Product Relationship Type Model — Optional Component Of

The value `Optional Component Of` stored in `Relationship Type` indicates that the entity labeled with one Product ID (e.g. `CVRFPID-0001`) is an optional component of an entity with another Product ID (e.g. `CVRFPID-0002`). These Product ID's SHOULD NOT be identical to provide minimal redundancy.

2.2.5 Product Tree Models

To model the relationships among products their hosting platforms, versions thereof, and more the CSAF CVRF Product Tree allows for the description of various topological configurations both to ease mapping real-world scenarios and to minimize data duplication.

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.

Four different configuration classes are supported: **Flat**, **Branched**, **Concatenated**, and **Grouped**.

Non-normative comment:

These classes are shortly described and motivated in the following subsections.

2.2.5.1 Product Tree Model — Flat

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

2.2.5.2 Product Tree Model — Branch

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. Every open **Branch** element is terminated with the actual product item in the form of a **Full Product Name** element.

2.2.5.3 Product Tree Model — 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.

2.2.5.4 Product Tree Model — 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.

2.2.6 Publisher Type Model

« One of the following five categories MUST be used as values for **Publisher** fields in CSAF CVRF exactly as written:

```
Coordinator
Discoverer
Other
User
Vendor
```

» [CSAF-2.2.6-1]

For usage context see example for usage of Document Publisher attribute Type.

Non-normative comment:

In the following sub sections these five terms are presented in lexical order and annotated to foster interoperable usage and as support for the reader.

2.2.6.1 Publisher Type Model — Coordinator

The value `Coordinator` of `Publisher` fields indicates 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.

2.2.6.2 Publisher Type Model — Discoverer

The value `Discoverer` of `Publisher` fields indicates individuals or organizations that find vulnerabilities or security weaknesses. This includes all manner of researchers.

2.2.6.3 Publisher Type Model — Other

The value `Other` of `Publisher` fields indicates a catchall for everyone else. Currently this includes editors, reviewers, forwarders, republishers, language translators, and miscellaneous contributors.

2.2.6.4 Publisher Type Model — User

The value `User` of `Publisher` fields indicates anyone using a vendor's product.

2.2.6.5 Publisher Type Model — Vendor

The value `Vendor` of `Publisher` fields indicates 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.

2.2.7 Reference Type Model

In the scope of CSAF CVRF, a `Reference` holds any reference to conferences, papers, advisories, and other resources that are related and considered related to either a surrounding part of or the entire document and to be of value to the document consumer.

The reference SHOULD contain a compartmentalized textual discussion constrained by its Type attribute.

« Any value for the **Type** attribute of any CSAF CVRF Reference element (regardless of its parent element) MUST be one of the following two categories:

```
External
Self
```

» [CSAF-2.2.7-1]

A Note may also be annotated with a `Title` and an `Audience` attribute. These optional attributes are intended 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 of the note, whereas Audience will indicate who is intended to read it.

Non-normative comment:

In the following sub sections the terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.7.1 Reference Type Model — External

The value `External` of `Reference Type` fields indicates, that this document is an external reference to a document or vulnerability in focus (depending on scope).

2.2.7.2 Reference Type Model — Self

The value `Self` of `Reference Type` fields indicates, that this document is a reference to this same document or vulnerability (also depending on scope).

2.2.8 Status Type Model

« The following three **Status** categories MUST be used to enumerate a status inside a CSAF CVRF document exactly as written:

```
    Draft
    Interim
    Final
```

» [CSAF-2.2.8-1]

For usage context see example for usage of Status element inside Document Tracking element.

In the following sub sections the terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.8.1 Status Type Model — Draft

The value `Draft` of `Status Type` fields indicates, that this is a 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.

2.2.8.2 Status Type Model — Final

The value `Final` of `Status Type` fields indicates, that the issuing party asserts the content is unlikely to change.

2.2.8.3 Status Type Model — Interim

The value `Interim` of `Status Type` fields indicates, that the issuing party asserts the content is unlikely to change. "Final" status is an indication only, and does not preclude updates.

2.2.9 Version Type Model

In CSAF CVRF versioning is represented with a simple hierarchical counter model. This is a numeric tokenized field of the format "nn" – "`nn.nn.nn.nn`".

To label change, it may be incremented in either major (first field i.e. left-most component) or minor (second field i.e. second-to-left component) notation to denote clearly the evolution of the content of the document. The third and fourth number slot is conventionally interpreted as patch version and build number, i.e. with ever decreasing relevance for external interfaces.

« Issuing parties MUST ensure that this field is incremented appropriately, even for the editorial or grammatical changes. » [CSAF-2.2.9-1]

« The value MUST be completely matched by the following regular expression:

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

» [CSAF-2.2.9-2]

Examples can be found in section 4.5.3 Document Tracking – Version.

2.2.10 Vulnerability CVE Type Model

Vulnerability measures given as defined in the MITRE standard Common Vulnerabilities and Exposures (CVE) model and are expected to be in a specific form to enhance interoperability.

« The `CVE` value MUST be completely matched by the following regular expression:

```
    CVE-[0-9\-]+
```

» [CSAF-2.2.10-1]

Non-normative comment:

The MITRE standard CVE tracking number for vulnerability naming provides improved tracking of vulnerabilities over time across different reporting sources. For more information about CVE cf. [CVE]

2.2.11 Vulnerability CVSS Version 2 Type Model

The calculations of the numerical CVSS version 2 scores are out of scope for this document. Constraints on the possible values are mapped as follows:

« The `BaseScoreV2`, `TemporalScoreV2`, and `EnvironmentalScoreV2` values MUST be single decimal values in the interval `[0.0, 10.0]` as enforced by the external CVSSv2 schema and thus must be elements of the following finite ordered set with `101` elements:

```
{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1.0, ... 9.9, 10.0}
```

» [CSAF-2.2.11-1]

The `cvssVectorV2` value is a string which can be shorter than the version 3 counterpart (only up to 76 characters). This string encapsulates all input for CVSS score calculation and SHOULD be a valid CVSS vector. Note: This score offers no version prefix in CVSS v2.

« The `cvssVectorV2` value MUST abide to the following length constraint:

```
length < 77 characters
```

» [CSAF-2.2.11-2]

The specific notation is expected to 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 in CSAF CVRF.

Non-normative comment:

The Common Vulnerability Scoring System version 2 (CVSSv2) aggregation for vulnerabilities provides uniform grading and improved tracking of vulnerabilities over time across different reporting sources. For more information about CVSS version 2 cf. [CVSS2]

2.2.12 Vulnerability CVSS Version 3 Type Model

The calculations of the numerical CVSS version 3 scores are out of scope for this document. Constraints on the possible values are mapped as follows:

« The `BaseScoreV3`, `TemporalScoreV3`, and `EnvironmentalScoreV3` values MUST be single decimal values in the interval `[0.0, 10.0]` as enforced by the external CVSSv3 schema and thus must be elements of the following finite ordered set with `101` elements:

```
{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1.0, ... 9.9, 10.0}
```

» [CSAF-2.2.12-1]

The `cvssVectorV3` value is a string which can be longer than the version 2 counterpart (up to `133` characters, where 2 characters have been added to the theoretical 131 characters of such a vector to account for newline characters on any platform). This string encapsulates all input for CVSS score calculation.

« The `cvssVectorV3` value MUST abide to the following length constraint ():

```
length < 133 characters
```

» [CSAF-2.2.12-2]

The specific notation is expected to follow the guidelines set forth in the CVSS v3 documentation at [CVSS3] (cf. section “Vector String” pp.17,18 there).

Note the 133-character limitation in CSAF CVRF to accommodate for the maximal length of 131 characters as derived from [CVSS3] plus accommodation for an added line end character notion for any platform.

Non-normative comment:

> The Common Vulnerability Scoring System version 3 (CVSSv3) aggregation for vulnerabilities provides uniform grading and improved tracking of vulnerabilities over time across different reporting sources. For more information about CVSS version 3 cf. [CVSS3]

2.2.13 Vulnerability CWE Type Model

Vulnerability measures given as defined in the Common Weakness Enumeration (CWE) model are expected to be in a specific form to enhance interoperability.

« Any `CWE` value MUST be completely matched by the following regular expression:

```
CWE-[1-9]\d{0,5}
```

» [CSAF-2.2.13-1]

Non-normative comment:

> The CWE number for weakness enumeration provides improved tracking of weaknesses over time across different reporting sources. For more information about CWE cf. [CWE].
>
> Citing from these MITRE CVE pages:
> "[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."

2.2.14 Vulnerability ID Type Model

The Vulnerability ID gives the producer a place to publish a unique label or tracking ID for the vulnerability (if such information exists) that is part of the security advisory published in the CSAF CVRF document.

« An `ID` value MUST be an alphanumeric token. » [CSAF-2.2.14-1]

Non-normative comments:

> 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 but is not to be used to publish the CVE tracking numbers (MITRE standard Common Vulnerabilities and Exposures), as these are specified inside the dedicated CVE element.

2.2.15 Vulnerability Involvement Type Model

The vulnerability involvement attribute `Status` indicates the involvement or engagement of a party in the vulnerability identification, scoping, and remediation process.

« The following categories MUST be used to enumerate a **Vulnerability Involvement** `Status` attribute inside a CSAF CVRF document:

```
Completed
Contact Attempted
Disputed
In Progress
```

```
Not Contacted
Open
```

» [CSAF-2.2.15-1]

Non-normative comments:

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).

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 CSAF CVRF documents at each state, or simply omit this element altogether. It is recommended, however, that vendors that issue CSAF CVRF documents indicating an open or in-progress Involvement should eventually expect to issue a document as Disputed or Completed.

In the following sub sections these six terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.15.1 Vulnerability Involvement Type Model — 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.

2.2.15.2 Vulnerability Involvement Type Model — Contact Attempted

The document producer attempted to contact the affected vendor.

2.2.15.3 Vulnerability Involvement Type Model — 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.

2.2.15.4 Vulnerability Involvement Type Model — 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.

2.2.15.5 Vulnerability Involvement Type Model — Not Contacted

The document producer has not attempted to make contact with the   affected vendor.

2.2.15.6 Vulnerability Involvement Type Model — 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.

2.2.16 Vulnerability Product Affected Status Type Model

To express the status of any product w.r.t. to being affected by a vulnerability, `Status` elements contain communicate this via a controlled vocabulary.

This `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.

« The following categories MUST be used to enumerate a **Vulnerability Product Affected Status** Type attribute inside a CSAF CVRF document:

```
    First Affected
    Known Affected
    Known Not Affected
    First Fixed
    Fixed
    Recommended
    Last Affected
```

» [CSAF-2.2.16-1]

In the following sub sections these seven terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.16.1 Vulnerability Product Affected Status Type Model — First Affected

This is first version of the affected release known to be affected by the vulnerability.

2.2.16.2 Vulnerability Product Affected Status Type Model — Known Affected

This version is known to be affected by the vulnerability.

2.2.16.3 Vulnerability Product Affected Status Type Model — Known Not Affected

This version is known not to be affected by the vulnerability.

2.2.16.4 Vulnerability Product Affected Status Type Model — First Fixed

This version contains the first fix for the vulnerability but may not be the recommended fixed version.

2.2.16.5 Vulnerability Product Affected Status Type Model — Fixed

This version contains a fix for the vulnerability but may not be the recommended fixed version.

2.2.16.6 Vulnerability Product Affected Status Type Model — Recommended

This version has a fix for the vulnerability and is the vendor-recommended version for fixing the vulnerability.

2.2.16.7 Vulnerability Product Affected Status Type Model — 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.

2.2.17 Vulnerability Remediation Type Model

Specific details on how to handle (and presumably, fix) a vulnerability are communicated via **Remediation** containers inside a CSAF CVRF document and a controlled vocabulary is defined for values of the associated Type attribute.

A **Remediation** container can be tied to one or more specific products by referencing these products using either the **Product ID** or other means. If the **Remediation** is meant to be general or nonspecific for all products, any **Product ID** or other ID's are to be omitted.

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

« The following categories MUST be used to enumerate a **Vulnerability Remediation** Type attribute inside a CSAF CVRF document:

```
Workaround
Mitigation
Vendor Fix
None Available
Will Not Fix
```

» [CSAF-2.2.17-1]

In the following sub sections these five terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.17.1 Vulnerability Remediation Type Model — 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.

2.2.17.2 Vulnerability Remediation Type Model — 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.

2.2.17.3 Vulnerability Remediation Type Model — 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.

2.2.17.4 Vulnerability Remediation Type Model — None Available

Currently there is no fix available. Description should contain details about why there is no fix.

2.2.17.5 Vulnerability Remediation Type Model — 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, declared end-of-life, or otherwise deprecated. Description should contain details about why there will be no fix issued.

2.2.18 Vulnerability Threat Type Model

To further detail the threat type over the evolution of a product vulnerability **Threat** containers can be tied to one or more specific products by referencing these products using either the **Product ID** or other ID’s. If the **Threat** is meant to be general or nonspecific for all products, the **Product ID** and other ID’s are to be omitted. The `Type` attribute takes its values from a proven controlled vocabulary.

« The following categories MUST be used to enumerate a **Vulnerability Threat** `Type` attribute inside a CSAF CVRF document:

```
Impact
Exploit Status
Target Set
```

» [CSAF-2.2.18-1]

In the following sub sections these three terms are presented in lexical order and annotated to support interoperable usage and as support for the reader.

2.2.18.1 Vulnerability Threat Type Model — Impact

Impact contains an assessment of the impact on the user or the target set if the vulnerability is successfully exploited. 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.

2.2.18.2 Vulnerability Threat Type Model — 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.”

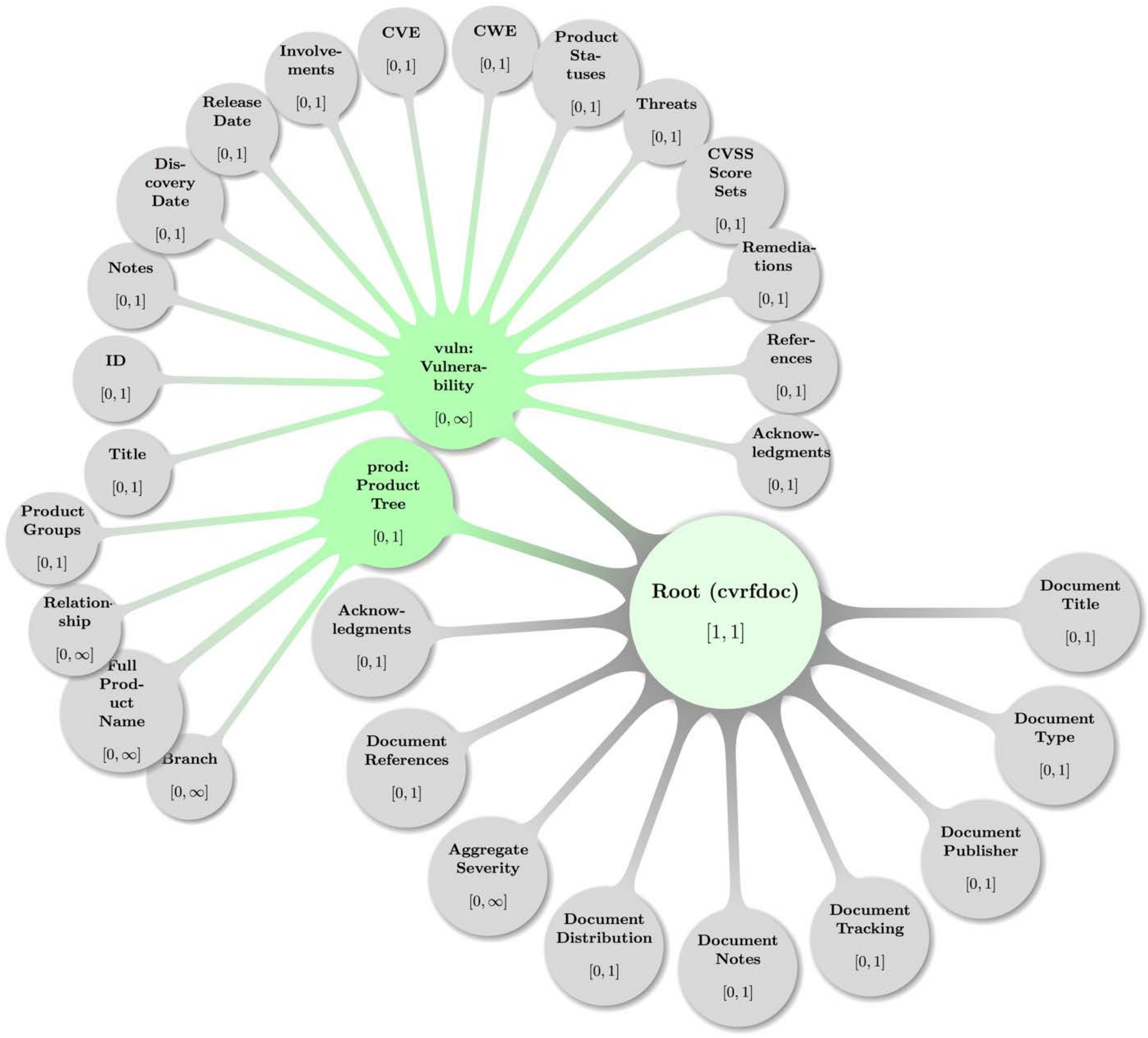
2.2.18.3 Vulnerability Threat Type Model — Target Set

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.

3 CSAF CVRF Model Tree Map

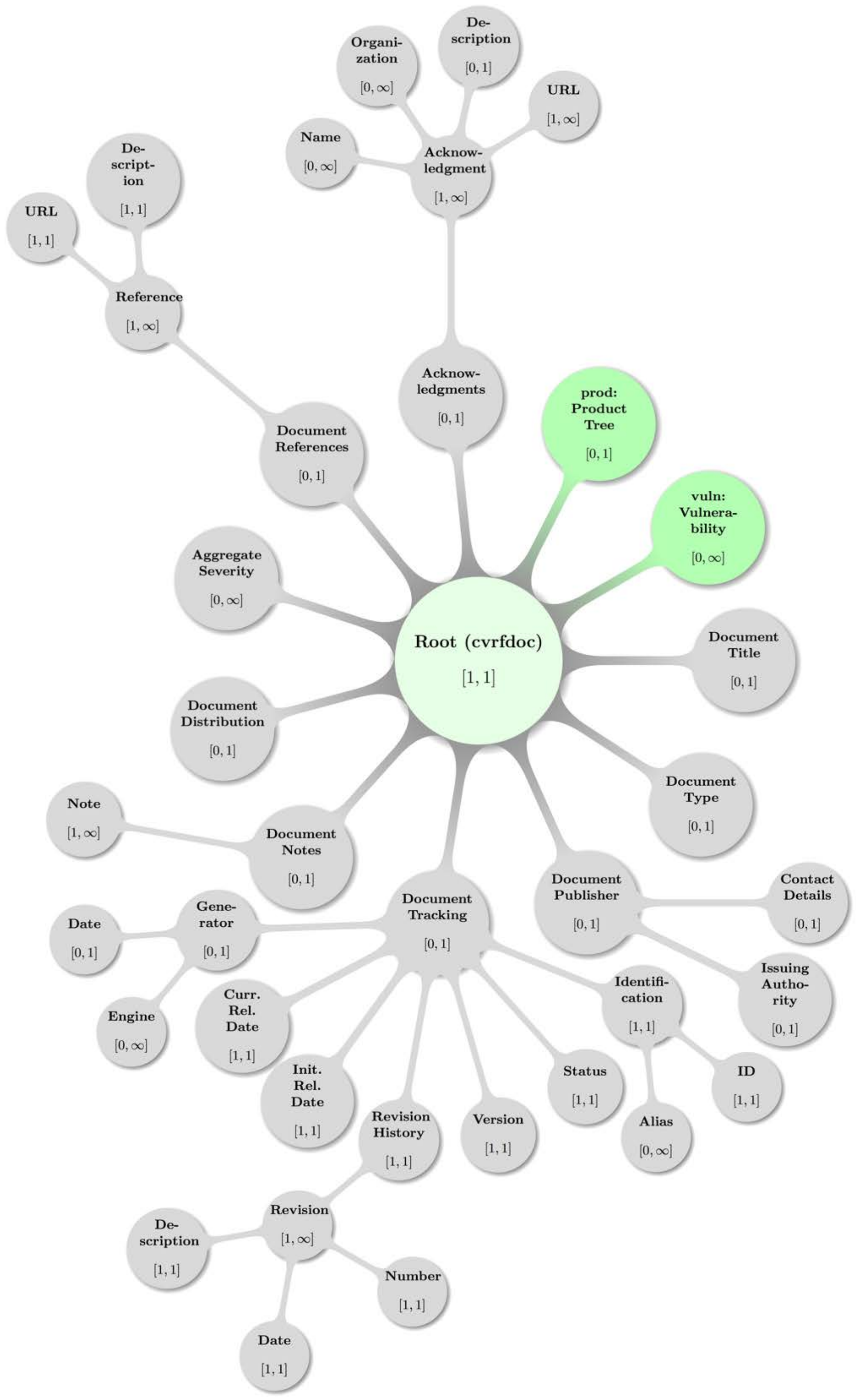
To assist navigating the topology of the CSAF CVRF version 1.2 document schema, a graphical tree rendering of the parent-child-grandchild relations among the elements under the single `cvrf:cvrfdoc` root is provided in the following diagram, where children and grandchildren (inside the `prod` and `vuln` namespaces) are displayed as they relate to the root element of a CSAF CVRF document:

Figure 1: ***CSAF CVRF Document Root*** *(`cvrf:cvrfdoc`) with children and grandchildren.*



Next a map of some abstract but specific and valid **CSAF CVRF Document** configuration emphasizing the topology of the elements from the `cvrf` namespace:

*Figure 2: A topologically valid **CSAF CVRF Document Root** configuration.*



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.

4 Document (Context) Schema Elements

« The nine top-level elements are defined in the `cvrf` XML schema file and if given MUST appear in the order listed below and as children of the `cvrf:cvrfdoc` single root element. » [CSAF-4-1]

These main constituents in sequence (Format is "**Concept**: `namespace:Element`") are:

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

Element `cvrf:cvrfdoc`

« The `cvrf:cvrfdoc` element is the root element of a CSAF CVRF Document and MUST contain the following child elements `cvrf:DocumentTitle`, `cvrf:DocumentType`, `cvrf:DocumentPublisher`, and `cvrf:DocumentTracking` all exactly once and in that order. » [CSAF-4.1-1]

« Following these child elements it MUST contain the elements `cvrf:DocumentNotes`, `cvrf:DocumentDistribution`, `cvrf:AggregateSeverity`, `cvrf:DocumentReferences`, `cvrf:Acknowledgements`, and `prod:ProductTree` all zero or once and in that order. » [CSAF-4.1-2]

« It MUST finally contain zero or more `vuln:Vulnerability` elements. » [CSAF-4.1-3]

4.2 Document Title

Element `cvrf:DocumentTitle`

« The `cvrf:DocumentTitle` element is required exactly once as first child of `cvrf:cvrfdoc` and its sole content MUST be a non-empty string. » [CSAF-4.2-1]

This string SHOULD be a definitive canonical name for the document, providing enough descriptive content to differentiate from other similar documents, ideally providing a unique "handle".

Non-normative Comment:

While this elements value – often just named “the title” – 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 expected document content.

If the document producer also publishes a human-friendly document that hand-in-hand with a CSAF 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.

Example 6:

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

Example 7:

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

Example 8:

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

Example 9:

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

Example 10:

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

Example 11:

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

Example 12:

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

4.3 Document Type

Element `cvrf:DocumentType`

« The `cvrf:DocumentType` element is required exactly once inside `cvrf:cvrfdoc` and its sole content MUST be a non-empty string. » [CSAF-4.3-1]

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.

Non-normative comment:

This type label is expected to be aligned with the content conveyed: If it is a “Security Advisory”, it should be named so, likewise a pure “Vulnerability Report” (see following below examples).

Example 13:

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

Example 14:

```
<DocumentType>Security Advisory</DocumentType>
```

Example 15:

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

4.4 Document Publisher

Element `cvrf:DocumentPublisher`

« The `cvrf:DocumentPublisher` element is required exactly once inside `cvrf:cvrfdoc`, it MUST provide the `Type` attribute. » [CSAF-4.4-1]

It MAY provide the `VendorID` attribute.

It MAY contain zero or one `cvrf:ContactDetails` element and zero or one `cvrf:IssuingAuthority` element.

« These MUST appear in the order `cvrf:ContactDetails` and `cvrf:IssuingAuthority` If both child elements are given. » [CSAF-4.4-2]

Attribute Type

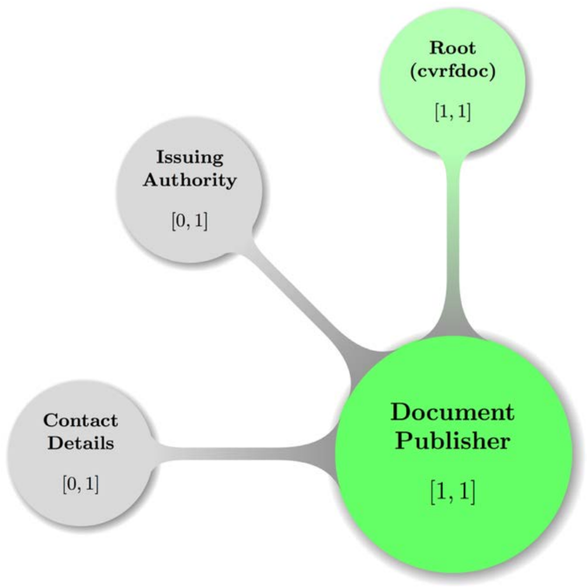
The value of `Type` is a token restricted by the set `cvrf-common:PublisherEnumType` defined in the `common.xsd` schema file and the values as given in section 2.2.2 Note Type Model.

Attribute `VendorID`

The value of `VendorID` is a string that SHOULD represent a unique identifier (OID) that a vendor uses as issued by FIRST under the auspices of IETF.

Map of some valid **Document Publisher** element level configuration including the parent node (**Root**):

*Figure 3: A topologically valid **Document Publisher** configuration.*



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.

Non-normative comment:

At the time of this writing, OID issuance by FIRST is still a work in progress, thus some samples are provided below, that use OID's from other standard MIB's.

Example 16:

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

Example 17: Cisco Systems Inc. OID in dot notation (cf. http://oid-info.com/get/1.3.6.1.4.1.9):

```
<DocumentPublisher Type="Vendor" VendorID="1.3.6.1.4.1.9"/>
```

Example 18: Cisco Systems Inc. via OID-IRI notation (slash-separated Unicode labels from root of OID tree):

```
<DocumentPublisher Type="Vendor" VendorID=" /ISO/Identified-
Organization/6/1/4/1/9"/>
```

Example 19: Cisco Systems Inc. MIB entry in ASN.1:

```
<DocumentPublisher Type="Vendor" VendorID="{iso(1) identified-organization(3)
dod(6) internet(1) private(4) enterprise(1) 9}"/>
```

4.4.1 Document Publisher – Contact Details

Element `cvrf:ContactDetails`

« The `cvrf:ContactDetails` element contains as its only content a non-empty string and MUST be present zero or one time inside `cvrf:DocumentPublisher` to convey author contact information such as address, phone number, or email. » [CSAF-4.4.1-1]

Example 20:

```
<ContactDetails>
  Name: Birgit Mustermensch\r\nOrganization: Internationale Sicherheit für Alle\r\n
  Phone Number: 004912345678901\r\nFax Number: 004912345678902\r\n
  Email Address: birgit.mustermensch@example.com
</ContactDetails>
```

4.4.2 Document Publisher – Issuing Authority

Element `cvrf:IssuingAuthority`

« The `cvrf:IssuingAuthority` element contains as its only content a non-empty string and MUST be used zero or once inside `cvrf:DocumentPublisher` to store the name of the issuing party and their authority to release the document, in particular, the party's constituency and responsibilities or other obligations. » [CSAF-4.4.2-1]

Non-normative comment:

This element is expected to also include instructions for contacting the issuer.

Example 21:

```
<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>
```

4.5 Document Tracking

Element `cvrf:DocumentTracking`

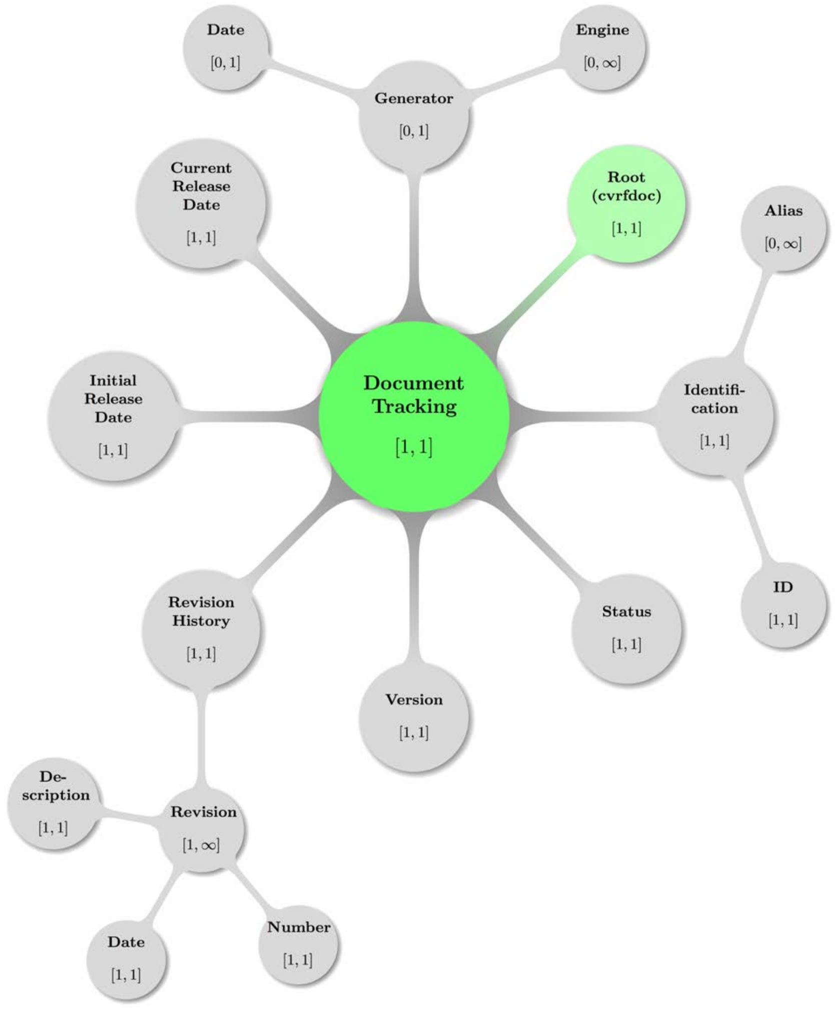
« The `cvrf:DocumentTracking` element required exactly once inside the `cvrf:cvrfdoc` root element and MUST contain the elements `cvrf:Identificaton`, `cvrf:Status`, `cvrf:Version`, `cvrf:RevisionHistory`, `cvrf:InitialReleaseDate`, and `cvrf:CurrentReleaseDate` all exactly once and in that order. » [CSAF-4.5-1]

« Following these child elements MUST be zero or one element `cvrf:Generator`. » [CSAF-4.5-2]

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

Following is the visual display of some valid **Document Tracking** configuration including the parent node `cvrf:cvrfdoc` (Document **Root**):

*Figure 4: A topologically valid **Document Tracking** configuration.*



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.

4.5.1 Document Tracking – Identification

Element `cvrf:Identification`

« The `cvrf:Identification` element is required exactly once inside the element `cvrf:DocumentTracking` and MUST contain the element `cvrf:ID` exactly once as first child. » [CSAF-4.5.1-1]

« Following this child element MUST be zero or more `cvrf:Alias` elements. » [CSAF-4.5.1-2]

The Document Tracking element cvrf:Identification is a container that holds all the identifiers for the CVRF document.

4.5.1.1 Document Tracking – Identification – ID

Element `cvrf:ID`

« The `crvf:ID` element MUST appear exactly once inside `cvrf:Identification` and its content MUST be a non-empty string that represents a short, unique identifier that allows to refer to the document unambiguously in any context. » [CSAF-4.5.1.1-1]

Its value refers to the condition of the document with regard to completeness and the likelihood of future editions.

The ID is a simple label. It is a string data type to provide for a wide range of numbering values, types, and schemes.

Its value SHOULD be assigned and maintained by the original document issuing authority.

Non-normative comment:

It is recommended that the ID contains monotonically increasing integer value parts, or is increasing in such a predictable manner that it does not contribute toward confusion or misinterpretation of numbering.

Common practice places a fixed producer acronym, the 4-digit year and a sequence integer number separated by e.g. dashes (-). So, for a fictitious producer Vendorix represented by the acronym VDX, and a security advisory number 42 produced in the calendar year 2017 this might result in an ID value of: `VDX-2017-42`

Careful consideration is required to ensure that construction of the ID does not contribute to confusion or collision with other labels.

4.5.1.2 Document Tracking – Identification – Alias

Element `cvrf:Alias`

« The `crvf:Alias` element MUST appear zero or more times inside `cvrf:Identification`. » [CSAF-4.5.1.2-1]

« If given every instance MUST contain a non-empty string that represents a distinct optional alternative ID used to refer to the document. » [CSAF-4.5.1.2-2]

Non-normative comment:

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.

4.5.2 Document Tracking – Status

Element `cvrf:Status`

« The `crvf:Status` element MUST appear exactly once in `cvrf:DocumentTracking` and its sole content MUST be a valid representative of the Status model documented in section 2.2.8 Status Type Model. » [CSAF-4.5.2-1]

Its value refers to the condition of the document with regard to completeness and the likelihood of future editions.

Non-normative comment:

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.

4.5.3 Document Tracking – Version

Element `cvrf:Version`

« The `crvf:Version` element MUST appear exactly once in `cvrf:DocumentTracking` and its sole content MUST be a valid representative of the Version model documented in section 2.2.9 Version Type Model. » [CSAF-4.5.3-1]

Example 22: Only major and minor version numbers stated:

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

Example 23: Major (1), minor (2) and patch (3) version numbers given:

```
<Version>1.2.3</Version>
```

Example 24: Build number 9876 appended to version triple (1.0.0):

```
<Version>1.0.0.9876</Version>
```

4.5.4 Document Tracking – Revision History

Element `cvrf:RevisionHistory`

« The `cvrf:RevisionHistory` element MUST be present exactly once inside `cvrf:DocumentTracking`, and MUST contain one or more `cvrf:Revision` elements. » [CSAF-4.5.4-1]

For every version or revision of the CSAF CVRF document, including the initial version it SHOULD hold matching `cvrf:Revision` elements.

4.5.4.1 Document Tracking – Revision History – Revision

Element `cvrf:Revision`

« The `cvrf:Revision` element MUST appear one or more times inside the element `cvrf:RevisionHistory`, and every instance MUST contain the elements `cvrf:Number`, `cvrf:Date`, and `cvrf:Description` all exactly once and in that order. » [CSAF-4.5.4.1-1]

The `cvrf:Revision` element contains all the information elements required to track the evolution of a CSAF CVRF document.

Non-normative comment:

Each change to a CSAF CVRF document should only be noteworthy, if accompanied by Number, Date, and Description information to be stored inside the child elements.

4.5.4.1.1 Document Tracking – Revision History – Revision – Number

Element `cvrf:Number`

« The `crvf:Number` element MUST appear exactly once in `cvrf:Revision` and its sole content MUST be a valid representative of the Version model documented in section 2.2.9 Version Type Model. » [CSAF-4.5.4.1.1-1]

Its value SHOULD contain the numeric version of the document.

The most recent `cvrf:Number` elements value should always match the value of the `cvrf:Version` element.

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).

Non-normative comment:

Examples of major or actionable 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

4.5.4.1.2 Document Tracking – Revision History – Revision – Date

Element `cvrf:Date`

« The element `cvrf:Date` MUST appear exactly once in `cvrf:Revision` and SHOULD record the date the revision was made and MUST be valid representative of the Date and Time model documented in section 2.2.1 Date and Time. » [CSAF-4.5.4.1.2-1]

4.5.4.1.3 Document Tracking – Revision History – Revision – Description

Element `cvrf:Description`

« The element `cvrf:Description` MUST appear exactly once in `cvrf:Revision` and SHOULD hold a single non-empty string representing a short description of the changes. » [CSAF-4.5.4.1.3-1]

Non-normative comment:

It can describe the conditions that prompted the change or be a short list of the items changed.

Example 25:

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

4.5.5 Document Tracking – Initial Release Date

Element `cvrf:InitialReleaseDate`

« The element `cvrf:InitialReleaseDate` MUST appear exactly once inside `cvrf:DocumentTracking` and SHOULD record the date that the document was initially released by the issuing party and MUST be valid representative of the Date and Time model documented in section 2.2.1 Date and Time. » [CSAF-4.5.5-1]

Example 26:

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

4.5.6 Document Tracking – Current Release Date

Element `cvrf:CurrentReleaseDate`

« The element `cvrf:CurrentReleaseDate` MUST appear exactly once inside `cvrf:DocumentTracking`, SHOULD be the current date that the document was released by the issuing party, and MUST be a valid representative of the Date and Time model documented in section 2.2.1 Date and Time. » [CSAF-4.5.6-1]

Example 27:

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

4.5.7 Document Tracking – Generator

Element `cvrf:Generator`

« The `cvrf:Generator` element MUST appear zero or once in `cvrf:DocumentTracking` and MUST contain the elements `cvrf:Engine` and `cvrf:Date` all zero or once and in that order. » [CSAF-4.5.7-1]

It 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.

4.5.7.1 Document Tracking – Generator – Engine

Element `cvrf:Engine`

« The optional `cvrf:Engine` element if present MUST be in `cvrf:Generator`. » [CSAF-4.5.7.1-1]

« Any instance MUST contain a non-empty string. » [CSAF-4.5.7.1-2]

This string SHOULD represent the name of the engine that generated the CSAF CVRF document, and MAY additionally refer to its version.

4.5.7.2 Document Tracking – Generator – Date

Element `cvrf:Date`

« The element `cvrf:Date` MUST appear zero or once in `cvrf:Generator`, SHOULD be the current date that the document was generated, and MUST be a valid representative of the Date and Time model documented in section 2.2.1 Date and Time Model. » [CSAF-4.5.7.2-1]

Because documents are often generated internally by a document producer and exist for a nonzero amount of time before being released, this field MAY be different from the Initial Release Date.

Example 28: Generator entry with fictitious engine and date given as date time with offset:

```
<Generator>
  <Engine>Magical Mitigation Machinery, version 1.2.3.42</Engine>
  <Date>2017-03-27T01:23:45+00:00</Date>
</Generator>
```

Example 29: Generator entry for another fictitious engine and date stated for AEST time zone (UTC+10) via offset

```
<Generator>
  <Engine>AnotherSSLVulnAdvisor xsslcsaf 0.9.987</Engine>
  <Date>2012-05-08T10:26:11+10:00</Date>
</Generator>
```

Example 30: Generator entry from existing vendor documentation and date given with time zone UTC (via Z token)

```
<Generator>
  <Engine>Red Hat rhsa-to-cvrf 1.0.1478</Engine>
  <Date>2012-05-08T10:26:11Z</Date>
</Generator>
```

Example 31: Full Document tracking element sample (with generator entry from previous example)

```
<DocumentTracking>
  <Identification><ID>RHSA-2010:0888</ID></Identification>
  <Status>Final</Status>
  <Version>1</Version>
  <RevisionHistory>
     <Revision>
       <Number>1</Number>
       <Date>2010-11-16T11:08:00Z</Date>
       <Description>Current version</Description>
     </Revision>
  </RevisionHistory>
  <InitialReleaseDate>2010-11-16T11:08:00Z</InitialReleaseDate>
  <CurrentReleaseDate>2010-11-16T11:08:00Z</CurrentReleaseDate>
  <Generator>
    <Engine>Red Hat rhsa-to-cvrf 1.0.1478</Engine>
    <Date>2012-05-08T10:26:11Z</Date>
  </Generator>
</DocumentTracking>
```

4.6 Document Notes

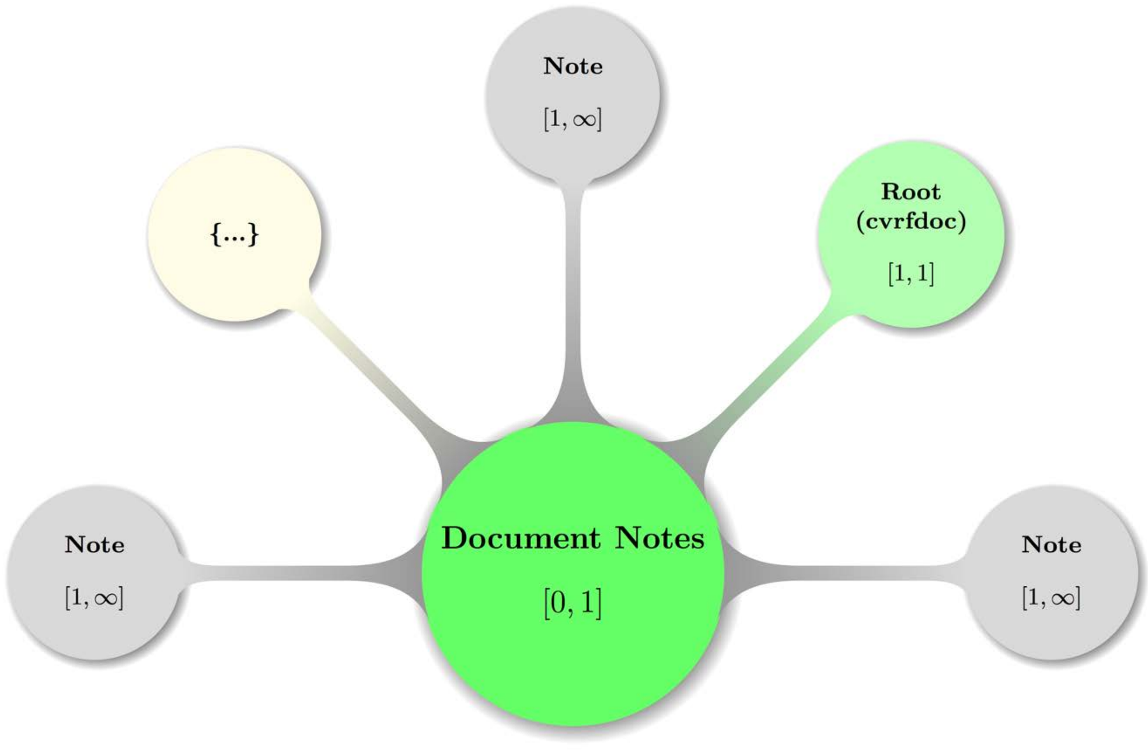
Element `cvrf:DocumentNotes`

« The `cvrf:DocumentNotes` element is an optional child of the document root element `cvrf:cvrfdoc` and if present MUST contain one or more `cvrf:Note` elements. » [CSAF-4.6-1]

It holds all of the document-level **Note** elements.

Following is a visual display of **Document Notes** including the parent element (**Document root**) in some valid configuration:

*Figure 5: A topologically valid **Document Notes** configuration.*



Again, 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 node carrying an ellipsis (…) shall hint at possible further **Note** elements.

4.6.1 Document Notes – Note

Element `cvrf:Note`

« The `cvrf:Note` element MUST occur one or more times inside the `cvrf:DocumentNotes` element. » [CSAF-4.6.1-1]

« Any instance MUST contain a non-empty string representing a **Note.** » [CSAF-4.6.1-2]

« It MUST provide a `Type` and an `Ordinal`. » [CSAF-4.6.1-3]

It MAY provide `Title` and `Audience` attributes.

The element `cvrf:Note` is a place to put all manner of text blobs related to the document as a whole. For further details cf. section 2.2.2 Note Type Model.

Attribute Title

The optional `Title` attributes value is a string, that SHOULD not be empty, and the value SHOULD be aligned with the annotations described in section 2.2.2 Note Type Model .

Attribute Audience

The optional `Audience` attributes value is a string, that SHOULD not be empty, and the value SHOULD be aligned with the annotations described in section 2.2.2 Note Type Model .

Attribute Type

« The required `Type` attribute MUST be a string with a valid enumeration value of the Note Type model as defined in section 2.2.2 Note Type Model . » [CSAF-4.6.1-4]

Attribute Ordinal

« The required `Ordinal` attribute MUST hold a **Positive Integer**. » [CSAF-4.6.1-5]

In addition to its domain the attribute value represents a mandatory, locally significant value used to track notes inside a CSAF CVRF document at the root (document) level.

Every `Ordinal` that tracks a `cvrf:Note` inside the `cvrf:Notes` is completely independent from an `Ordinal` tracking a `Note` in a different namespace e.g. inside `vuln:Notes`.

It is provided to uniquely identify a Note. There should be exactly one of these values per every `cvrf:Note` inside the `cvrf:Notes` container element.

The ascendingly ordered set of all such `Ordinal` values SHOULD be identical to the subset of Positive Integers smaller or equal to the length of the set.

Non-normative comment:

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.

Example 32: A `cvrf:Note` with all attributes provided and adhering to value convention for `Ordinal`:

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

4.7 Document Distribution

Element `cvrf:DocumentDistribution`

« The `cvrf:DocumentDistribution` element MUST be present zero times or once as a child of `cvrf:cvrfdoc` and its value MUST be a non-empty string. » [CSAF-4.7-1]

It should contain details about constraints, if any, for sharing the CSAF CVRF document with additional recipients.

These constraints MAY include instructions on how to reproduce, share, copy, or otherwise distribute the document.

Example 33:

```
<DocumentDistribution xml:lang="de">
  Urheberrechtlich geschützt, 2017, Fiktive GmbH.
  Keine Weitergabe ohne vorherige schriftliche Zustimmung;
  Anzufragen via E-Mail unter cert@fg.example.com
  Quelle für dieses Dokument: https://fg.example.com/sa/fg-sa-2017-123
</DocumentDistribution>
```

Example 34:

```
<DocumentDistribution xml:lang="en">
  Copyright © 2016 Previous Year Again, Inc. All rights reserved.
</DocumentDistribution>
```

4.8 Aggregate Severity

Element `cvrf:AggregateSeverity`

« The optional `cvrf:AggregateSeverity` element if present MUST be in `cvrf:cvrfdoc`. » [CSAF-4.8-1]

« Any instance MUST contain a non-empty string. » [CSAF-4.8-2]

It MAY provide a `Namespace` attribute.

The element `cvrf:AggregateSeverity` is a vehicle that is provided by the document producer to convey the urgency and criticality with which the one or more vulnerabilities reported 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.

Attribute Namespace

The optional `Namespace` attribute’s value SHOULD be a URL (`xs:anyURI`) pointing to the namespace so referenced.

Non-normative comment:

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).

Example 35:

```
<AggregateSeverity Namespace="https://example.com/se/c/">Important</AggregateSeverity>
```

4.9 Document References

Element `cvrf:DocumentReferences`

« The optional `cvrf:DocumentReferences` element if present MUST occur as if appended to the position that a `cvrf:AggregateSeverity` element will take inside the `cvrf:cvrfdoc root` element. » [CSAF-4.9-1]

« If present, `cvrf:DocumentReferences` MUST contain [1, ∞] instances of the element `cvrf:DocumentReference`. » [CSAF-4.9-2]

The `cvrf:DocumentReferences` element is a container that SHOULD provide a place in its children `cvrf:DocumentReference` elements to include references to any conferences, papers, advisories, and other resources that are related and considered to be of value to the document consumer.

4.9.1 Document References – Reference

Element `cvrf:Reference`

« The `cvrf:Reference` element MUST occur one or more times inside the element `cvrf:References`. » [CSAF-4.9.1-1]

« Any instance MUST contain the elements `cvrf:URL` and `cvrf:Description` all exactly once and in that order. » [CSAF-4.9.1-2]

« It MUST provide either a `Type` attribute or a default will be taken instead. » [CSAF-4.9.1-3]

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

Attribute Type

« The required `Type` attribute is if not present understood per default as the enumeration value `External` or if given MUST be a string with a valid enumeration value of the Reference Type model as defined in section 2.2.7 Reference Type Model. » [CSAF-4.9.1-4]

The Type attribute denotes the type of the document reference relative to the given document and as described in section 2.2.7 Reference Type Model.

4.9.1.1 Document References – Reference – URL

Element `cvrf:URL`

« The `cvrf:URL` element MUST be present exactly once in `cvrf:Reference` and its content SHOULD be the fixed URL (`xs:anyURI`) or location of reference. » [CSAF-4.9.1.1-1]

4.9.1.2 Document References – Reference – Description

Element `cvrf:Description`

« The `cvrf:Description` element MUST occur exactly once in `cvrf:Reference`. » [CSAF-4.9.1.2-1]

« Any instance MUST contain a non-empty string representing the description of the related document**.** » [CSAF-4.9.1.2-2]

Non-normative comment:

This can be a descriptive title or the name of the referenced artifact.

Example 36:

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

4.10 Acknowledgements

Element `cvrf:Acknowledgements`

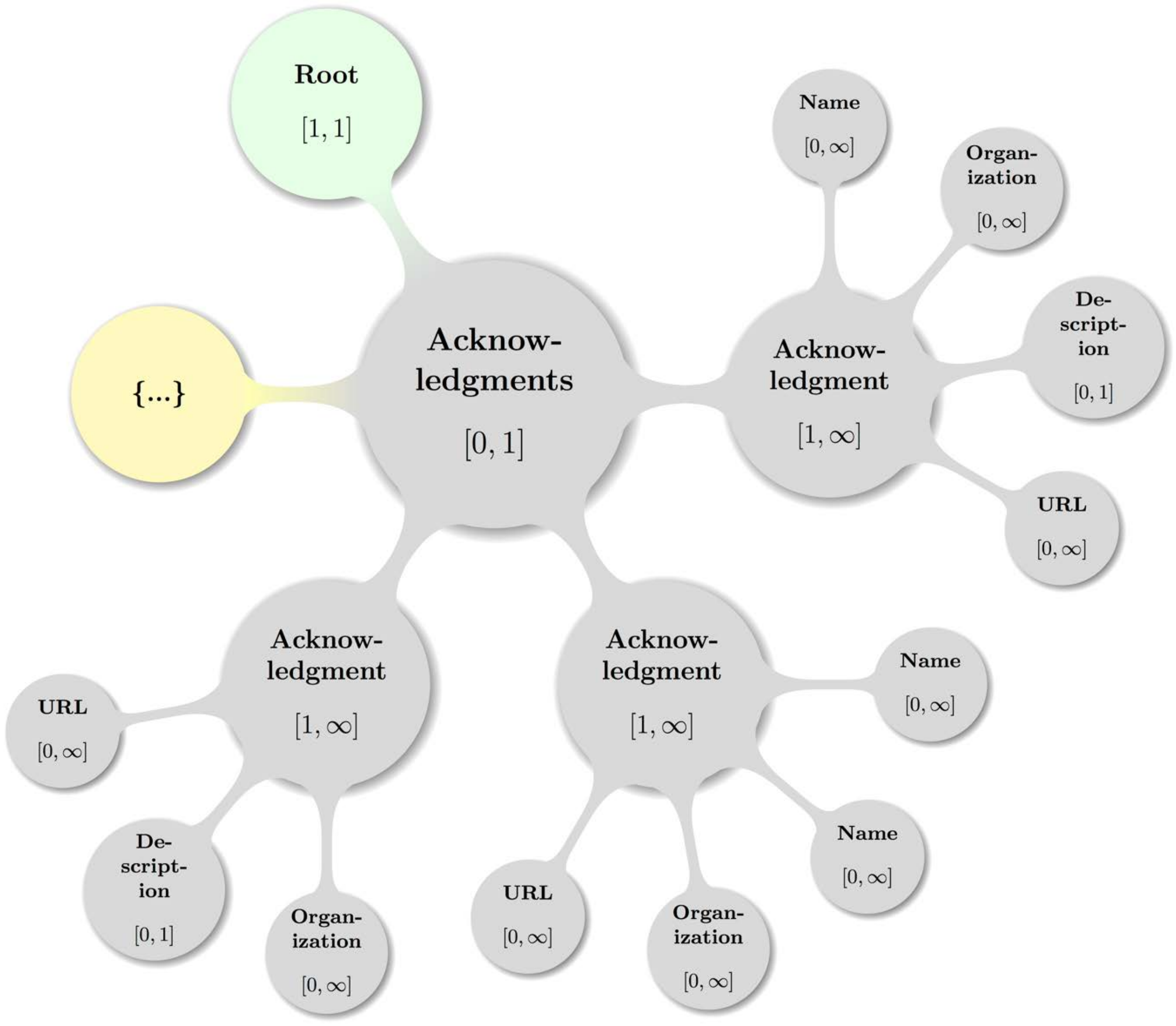
« The optional `cvrf:Acknowledgements` element if present MUST be in `cvrf:cvrfdoc`. » [CSAF-4.10-1]

« An instance MUST contain one or more `cvrf:Acknowledgement` elements**.** » [CSAF-4.10-2]

The `cvrf:Acknowledgements` element is a container that SHOULD provide a place in the children `cvrf:Acknowledgement` elements to note recognition of external parties.

Following is a visual representation of some valid **Document Acknowledgements** configuration including the parent node `cvrf:cvrfdoc` (**Document Root**) — again with the node labeled {…} indicating further possible **Acknowledgement** subtrees:

*Figure 6: A topologically valid **Document Acknowledgements** configuration.*



Non-normative comment:

This is the direct `cvrf:cvrfdoc` child element. For a sample display of the **Acknowledgements** container of a Vulnerability element cf. section 6.15).

4.10.1 Acknowledgements – Acknowledgement

Element `cvrf:Acknowledgement`

« The `cvrf:Acknowledgement` element MUST occur one or more times inside the `cvrf:Acknowledgements` element. » [CSAF-4.10.1-1]

« Any instance MUST contain elements `cvrf:Name [0, ∞]`, `cvrf:Organization [0, ∞]`, `cvrf:Description [0, 1]`, and `cvrf:URL [0, ∞]` with the given **Cardinality Ranges** and in that order**.** » [CSAF-4.10.1-2]

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.

4.10.1.1 Acknowledgements – Acknowledgement – Name

Element cvrf:Name

« The cvrf:Name element MUST occur zero or more times in cvrf:Acknowledgement. » [CSAF-4.10.1.1-1]

« Any instance MUST contain a non-empty string, that SHOULD contain the name of the party being acknowledged. » [CSAF-4.10.1.1-2]

4.10.1.2 Acknowledgements – Acknowledgement – Organization

Element cvrf:Organization

« The cvrf:Organization element MUST be present zero or more times inside any cvrf:Acknowledgement element. » [CSAF-4.10.1.2-1]

«Any instance MUST contain a non-empty string, that SHOULD contain the organization of the party or if cvrf:Name is omitted, the organization itself that is being acknowledged representing the description of the related document. » [CSAF-4.10.1.2-2]

4.10.1.3 Acknowledgements – Acknowledgement – Description

Element cvrf:Description

« The cvrf:Description element MUST be present zero or one time inside any cvrf:Acknowledgement element. » [CSAF-4.10.1.3-1]

« Any instance MUST contain a non-empty string, that SHOULD represent any contextual details the document producers wish to make known about the acknowledgment or acknowledged parties. » [CSAF-4.10.1.3-2]

If attributing to multiple organizations, each contributor should be grouped with that Organization within a single Acknowledgment container.

Non-normative comment:

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.

4.10.1.4 Acknowledgements – Acknowledgement – URL

Element cvrf:URL

« The cvrf:URL element MUST be contained zero or more times per any cvrf:Acknowledgement element and its content SHOULD give the optional URL (xs:anyURI) to the person, place, or thing being acknowledged. » [CSAF-4.10.1.4-1]

Example 37:

```
<Acknowledgments>
  <Acknowledgment>
    <Name>Johann Sebastian Bach</Name>
    <Organization>Security Fugue LLC</Organization>
    <Description>First analysis of Coordinated Multi-Stream Attack (CMSA)</Description>
    <URL>https://secure-fugue.example.com/team/~jsb</URL>
  </Acknowledgment>
</Acknowledgments>
```

5 Product Tree Schema Elements

Product information in CSAF 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:Relationship`
4.	**Product Groups**:	`prod:ProductGroups`

» [CSAF-5-1]

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 information to aid the reader in creating or navigating the graph that can be spanned by instances of a **Product Tree**.

5.1 Product Tree

Element `prod:ProductTree`

« The optional `prod:ProductTree` element MUST occur with cardinality [0, 1] as child of `cvrf:cvrfdoc`. » [CSAF-5.1-1]

« If given, the instance MUST contain `prod:Branch` [0, ∞], `prod:FullProductName` [0, ∞], `prod:FullProductName` [0, ∞], `prod:ProductGroups` [0, 1], elements with noted cardinalities and in that order. » [CSAF-5.1-2]

The optional element `prod:ProductTree` is a container for all fully qualified product names that can be referenced elsewhere in the document.

References to be named specifically when describing the products that are affected by a vulnerability using the **Product Statuses**, **Vulnerability Threats**, **Vulnerability CVSS Score Sets**, and **Vulnerability Remediation** containers.

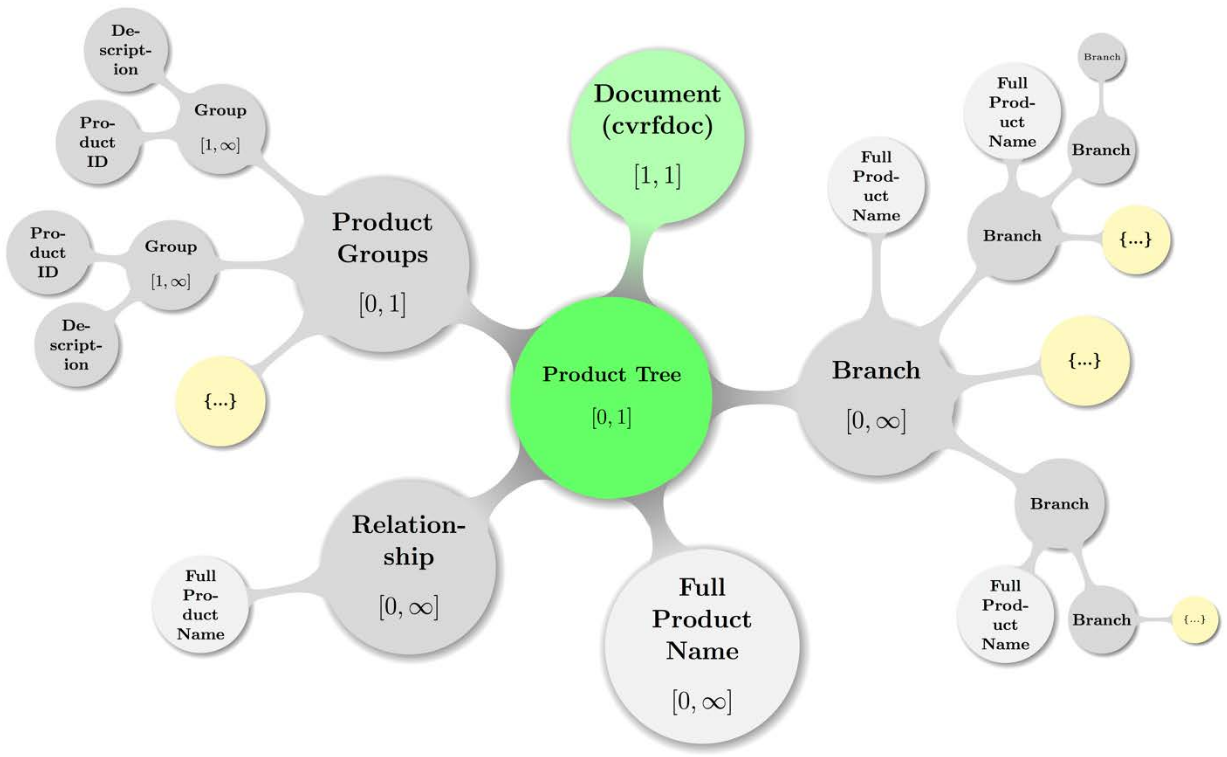
The **Product Tree** can have as many branches as needed, but every endpoint of the tree must be terminated with a **Full Product Name** element, which represents a product that can be referenced elsewhere.

Example 38:

```
<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>
```

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

*Figure 7: A topologically valid **Product Tree** configuration.*



Again, 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

Element prod:Branch

« The optional prod:Branch choice element MUST occur with cardinality [0, ∞] inside the prod:ProductTree element or nested inside other prod:Branch instances. » [CSAF-5.1.1-1]

« An instance MUST contain either a prod:Branch or a prod:FullProductName element. » [CSAF-5.1.1-2]

The prod:FullProductName element has no children (is thus always a terminating or leaf element) while a prod:Branch can either contain further children or terminate its tree branch.

Attribute Name

The mandatory Name attribute of the **Branch** element and its relation to the equally required Type attribute are documented in section 2.2.3 Product Branch Type Model.

Attribute Type

The required Type attribute of the **Branch** element is documented in section 2.2.3 Product Branch Type Model.

Non-normative comment:

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.

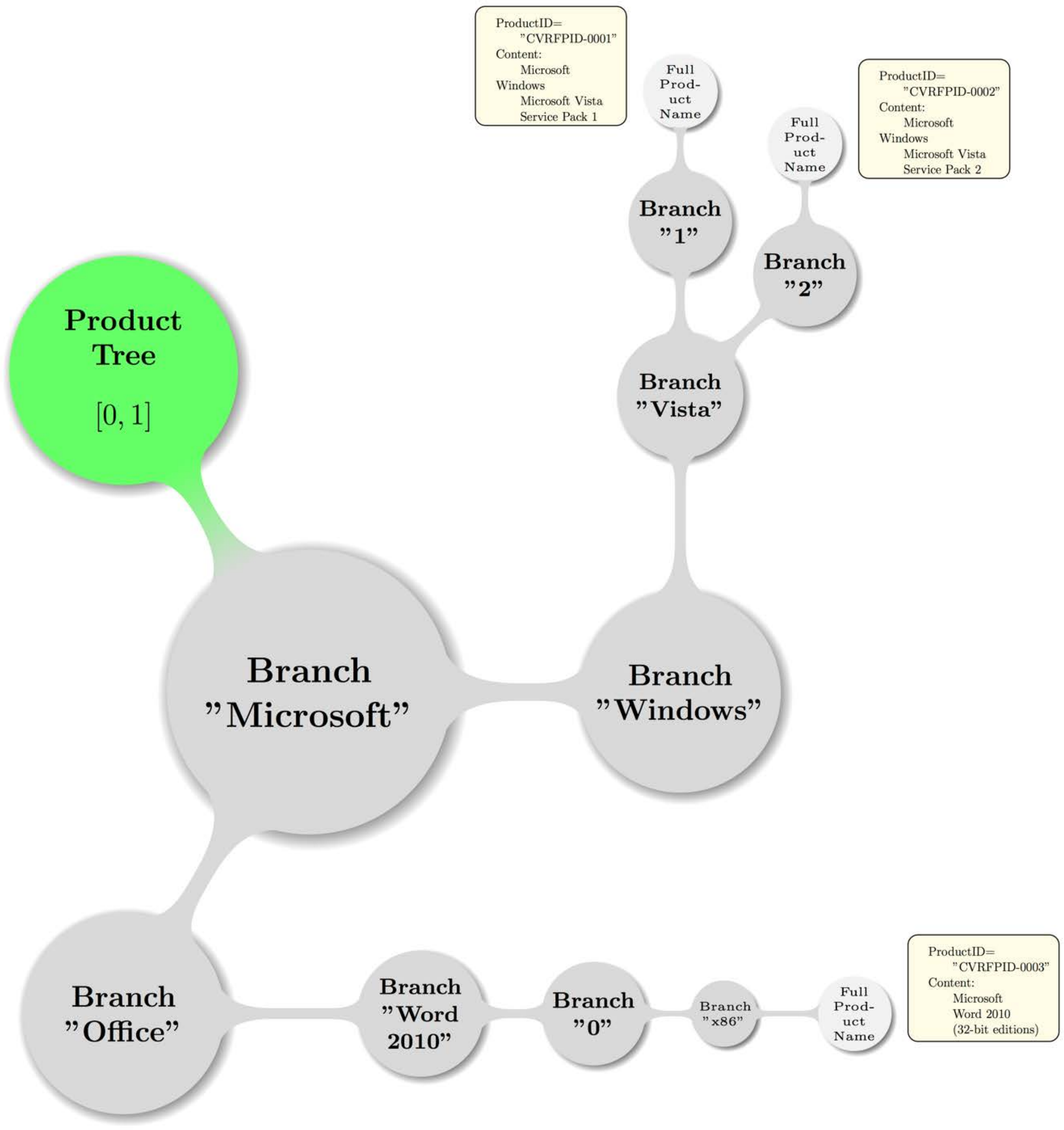
*Example 39: Nesting of **Branches** in a **Branch** subtree:*

```
<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 (**Error! Reference source not found.**).

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 8: A topologically valid **Branch** configuration*


ProductID= "CVRFPID-0001" Content: Microsoft Windows Microsoft Vista Service Pack 1
Full Product Name
ProductID= "CVRFPID-0002" Content: Microsoft Windows Microsoft Vista Service Pack 2
Full Product Name
Branch "1"
Branch "2"
Product Tree
[0, 1]
Branch "Vista"
Branch "Microsoft"
Branch "Windows"
Branch "Office"
Branch "Word 2010"
Branch "0"
Branch "x86"
Full Product Name
ProductID= "CVRFPID-0003" Content: Microsoft Word 2010 (32-bit editions)

5.1.2 Product Tree – Full Product Name

Element `prod:FullProductName`

« The `prod:FullProductName` element MUST be a child of cardinality [1, ∞] for all possible locations inside the product tree representation. » [CSAF-5.1.2-1]

This elements instances have multiple possible parent elements: `prod:ProductTree`, `prod:Releationship`, and `prod:Branch`.

The `prod:FullProductName` elements define the endpoints of the **Product Tree** and occur either 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.

Attribute `ProductID`

The `ProductID` attribute is a token 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 `ProductID` as long as it uniquely identifies a product in the context of the current document.

Attribute `CPE`

The (Common Platform Enumeration) `CPE` attribute refers to a method for naming platforms external to CSAF CVRF.

« The `CPE` attribute if present MUST have a value, that is a valid `cpe-lang:namePattern` as defined in the external specification [CPE23_N] and related schemas. » [CSAF-5.1.2-2]

Non-normative comment:

Examples for `ProductID` values include incremental integers or Globally Unique Identifiers (GUIDs).

The structure for CPE as required for a valid CSAF CVRF document is described in its specification documents ([CPE23_N] et al.).

In short: The `CPE` can be either an integer (if there exists a known entry for the platform in question) or a candidate string from the vendor if no commonly registered entry yet exists (at the NIST CPE registry site).

Example 40:

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

Example 41:

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

5.1.3 Product Tree – Relationship

Element `prod:Relationship`

« The `prod:Relationship` element MUST be present with cardinality [0, ∞] in `prod:Tree` and if given MUST contain one or more `prod:FullProductName` instances. » [CSAF-5.1.3-1]

The `prod: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.

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: `ProductReference` and `RelatesToProductReference` each contain the `ProductID` token for the two products that will form the relationship, and the `RelationType` attribute defines how the products are related.

Attribute `ProductReference`

The required `ProductReference attribute` contains the `ProductID` token of one of the two products that will form the relationship. For directed relationships the producer SHOULD associate correctly.

Attribute `RelationType`

The allowed values of the required `RelationType` attribute are of an enumeration type and as defined in 2.2.4 Product Relationship Type Model.

Attribute `RelatesToProductReference`

The required `RelatesToProductReference` attribute contains the `ProductID` token of the other of the two products that will form the relationship. Again: For directed relationships the producer SHOULD associate correctly.

Non-normative comment:

The situation where a need for declaring a Relationship arises, is given when a product is e.g. vulnerable only when installed together with another, or to describe operating system components.

Example 42:

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="Optional Component Of"
 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 43:

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="Installed On"
 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

Element `prod:ProductGroups`

« The `prod:ProductGroups` element MUST be present zero or one time inside a given `prod:ProductTree` element and if present MUST contain one or more `prod:Group` elements. » [CSAF-5.1.4-1]

The element `prod:ProductGroups` is a container, that defines whether **Full Product Name** elements in the product tree will be grouped into logical groups.

If groups are defined, products can be referred to using the `GroupID` 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.

Non-normative comment:

Given that a single product can be a member of more than one group, some areas of the CSAF CVRF document may prohibit product references by group to avoid ambiguity.

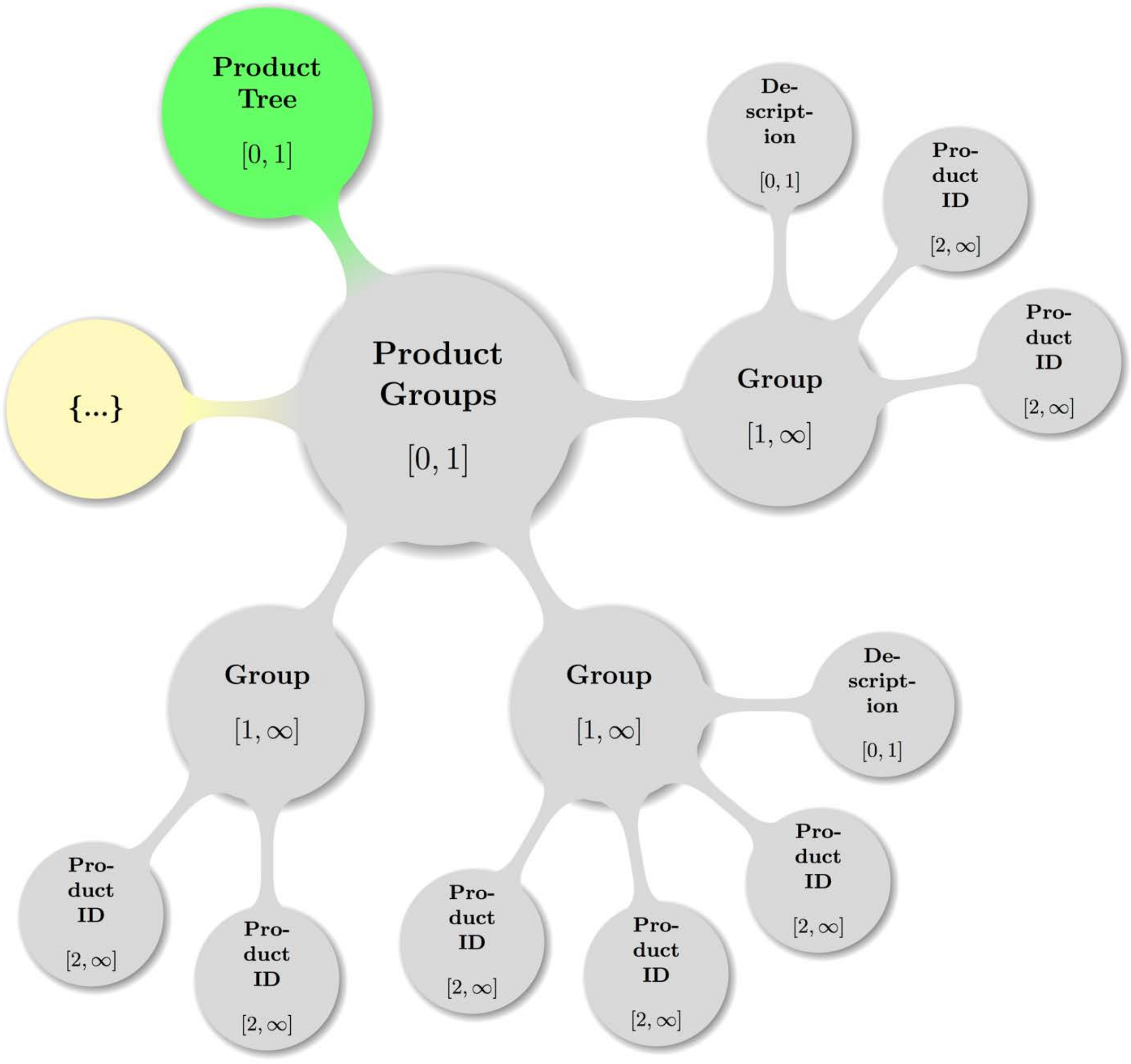
Example 44:

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>
```

A visual map of some fictitious sample **Product Groups** sub tree including the parent node (**Product Tree**) with the node labeled {…} indicating further possible **Group** subtrees is depicted below.

*Figure 9: A topologically valid **Product Groups** configuration*



5.1.4.1 Product Tree – Product Groups – Group

Element `prod:Group`

« The `prod:Group` element MUST be present one or more times as child of the optional `prod:ProductGroups` element and MUST contain zero or one `prod:Description` and 2 or more `prod:ProductID` elements and in that order. » [CSAF-5.1.4.1-1]

The element `prod: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 every member of the group.

Attribute `GroupID`

The `GroupID` 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 `GroupID` as long as it uniquely identifies a group in the context of the current document.

Non-normative comment:

Examples for `GroupID` attribute values include incrementing integers or GUIDs.

5.1.4.1.1 Product Tree – Product Groups – Group – Description

Element `prod:Description`

« The `prod:Description` element MUST be present zero or one time in `prod:Group` and if given is a short, optional description of the group. » [CSAF-5.1.4.1.1-1]

Example 45:

```
<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

Element `prod:ProductID`

« The `prod:ProductID` element MUST be present 2 or more times in `prod:Group` elements and every instance defines a member of a group by referring to the unique `ProductID` attribute of a **Full Product Name** element. » [CSAF-5.1.4.1.2-1]

Example 46:

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 CSAF 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 listed 14 second-level elements 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`

» [CSAF-6-1]

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.

6.1 Vulnerability

Element `vuln:Vulnerability`

« The `vuln:Vulnerability` element MUST be present zero more times in `cvrf:cvrfdoc` and MUST contain the following child elements `vuln:Title` [0, 1], `vuln:ID` [0, 1], `vuln:Notes` [0, 1], `vuln:DiscoveryDate` [0, 1], `vuln:ReleaseDate` [0, 1], `vuln:Involvements` [0, 1], `vuln:CVE` [0, 1], `vuln:CWE` [0, ∞], `vuln:ProductStatuses` [0, 1], `vuln:Threats` [0, 1], `vuln:CVSSScoreSets` [0, 1], `vuln:Remediation` [0, 1], `vuln:References` [0, 1], and `vuln:Acknowledgments` [0, 1] as per given cardinalities and in that order. » [CSAF-6.1-1]

The optional element `vuln:Vulnerability` is a container for the aggregation of all fields that are related to a single vulnerability in the document.

The cardinality allows to describe zero, one, or many vulnerabilities in a single CSAF CVRF document.

Attribute `Ordinal`

« The required attribute `Ordinal` is a locally significant value used to track vulnerabilities inside a CSAF CVRF document that MUST be an integer in the interval [1, ∞]. » [CSAF-6.1-2]

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).

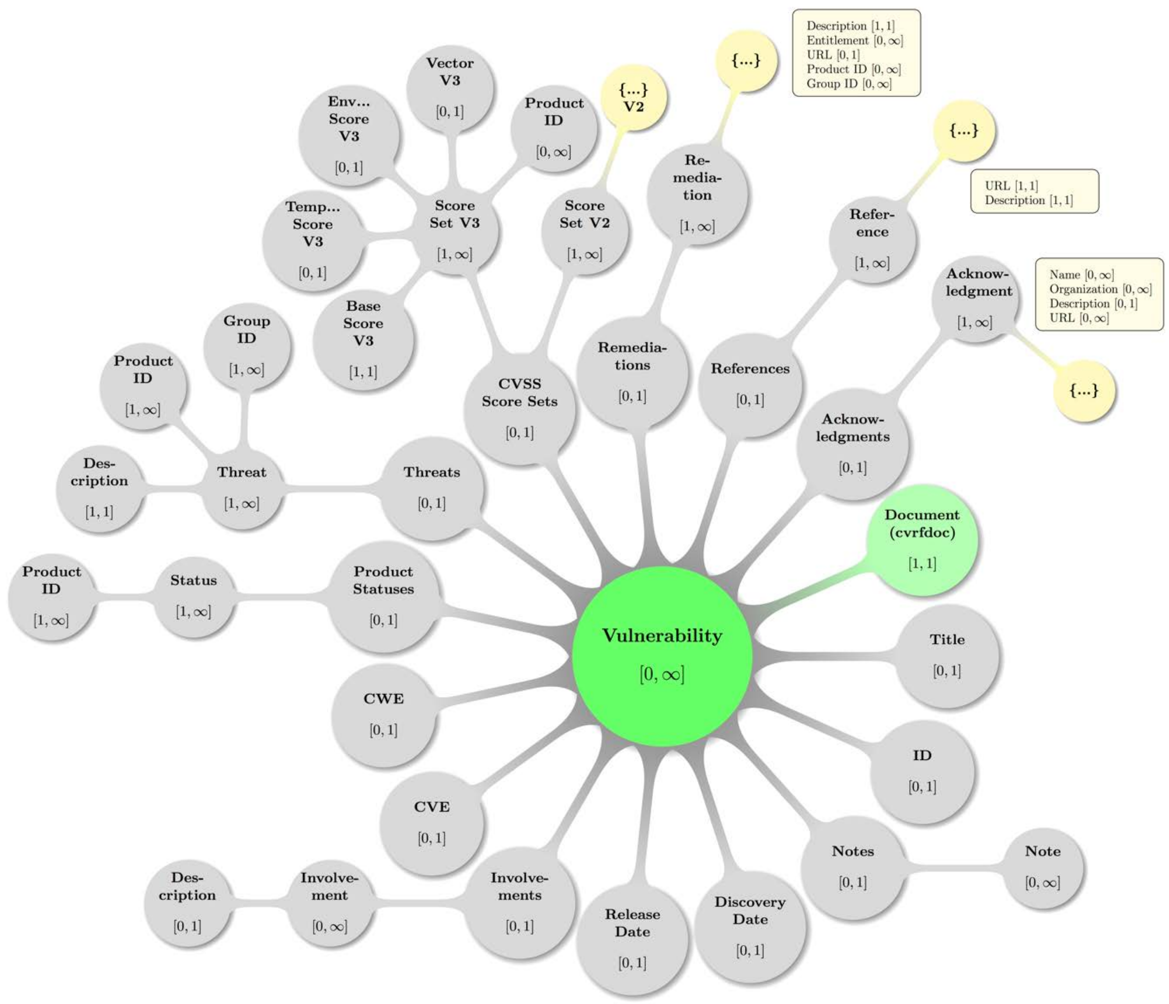
There SHOULD be one of these values for every **Vulnerability** container in a document, and it is recommended that `Ordinal` should be instantiated as a monotonically increasing counter, indexed from 1.

Example 47:

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

A visual map of **Vulnerability** element including the parent node (**Document**) in some valid configuration spanning multiple sub trees is given below.

*Figure 10: A topologically valid **Vulnerability** configuration.*



As before, 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

Element `vuln:Title`

« The `vuln:Title` element MUST be present zero or one time in `vuln:Vulnerability` and if present gives the document producer the ability to apply a canonical name or title to the vulnerability. » [CSAF-6.2-1]

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.**

Example 48:

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

6.3 Vulnerability – ID

Element `vuln:ID`

« The `vuln:ID` element MUST be present zero or one time in `vuln:Vulnerability` and if present gives the document producer a place to publish a unique label or tracking ID for the vulnerability (if such information exists). » [CSAF-6.3-1]

The value domain for `vuln:ID` elements is further documented in section 2.2.14 Vulnerability ID Type Model.

The attribute `SystemName` is required.

Attribute `SystemName`

The required attribute `SystemName` indicates the name of the vulnerability tracking or numbering system that this **ID** comes from.

Every **ID** value SHOULD have exactly one `SystemName`.

Non-normative comment:

It is helpful if document producers use unique and consistent system names.

Example 49:

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

6.4 Vulnerability – Notes

Element `vuln:Notes`

« The `vuln:Notes` element MUST be present zero or one time in `vuln:Vulnerability` and if present contain one or more `vuln:Note` elements.
» [CSAF-6.4-1]

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

6.4.1 Vulnerability – Notes – Note

Element `vuln:Note`

« The `vuln:Note` element MUST be present one or more times in `vuln:Notes` and is a place to put all manner of text blobs related to the vulnerability. » [CSAF-6.4.1-1]

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.

The vuln:Note element has four attributes, two of them are required: `Type` and `Ordinal`.

`Title` and `Audience` are the two optional attributes to give human readers context around what they are about to read.

Akin to the **Document Notes** element, the note SHOULD contain a compartmentalized textual discussion constrained by its `Type` attribute.

Attribute Type

« The value of the attribute `Type` MUST be one of the enumeration values as described in section 2.2.2 Note Type Model. » [CSAF-6.4.1-2]

Attribute `Ordinal`

`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.

Every `Ordinal` that tracks a **Note** inside **Vulnerability Notes** is completely independent from any `Ordinal` tracking a **Note** inside **Document Notes**.

Attribute `Title`

The optional attribute `Title` SHOULD be a concise description of what is contained in the text.

Attribute `Audience`

The optional attribute `Audience` SHOULD indicate who is intended to read it.

Example 50:

```
<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

Element `vuln:DiscoveryDate`

« The `vuln:DiscoveryDate` element MUST be present zero or one time inside any `vuln:Vulnerability` instance element and if given holds the date and time the vulnerability was originally discovered. » [CSAF-6.5-1]

All date like values in CSAF CVRF require a date and a time (cf. section 2.2.1 Date and Time).

Example 51:

```
<DiscoveryDate>2010-11-03T00:00:00Z</DiscoveryDate>
```

6.6 Vulnerability – Release Date

Element `vuln:ReleaseDate`

« The `vuln:ReleaseDate` element MUST be present zero or one time inside any `vuln:Vulnerability` instance element and if given holds the date and time the vulnerability was originally released into the wild. » [CSAF-6.6-1]

All date like values in CSAF CVRF require a date and a time (cf. section 2.2.1 Date and Time).

Example 52:

```
<ReleaseDate>2010-11-16T00:00:00Z</ReleaseDate>
```

6.7 Vulnerability – Involvements

Element vuln:Involvements

« The vuln:Involvements MUST be present zero or one time inside any vuln:Vulnerability element and if present contain one or more vuln:Involvement elements. » [CSAF-6.7-1]

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

Matching the possibly multiple **Involvements** containers, multiple parties can comment on their levels of involvement.

6.7.1 Vulnerability – Involvements – Involvement

Element vuln:Involvement

« The vuln:Involvment element MUST be present one or more times in vuln:Involvements and MUST contain zero or one vuln:Description elements. » [CSAF-6.7.1-1]

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 two attributes Party and Status are both required.

Attribute Party

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 value of the attribute Party MUST be as defined in section 2.2.6 Publisher Type Model. » [CSAF-6.7.1-2]

Attribute Status

« The attribute Status indicates the level of involvement of (the) Party and the enumeration value MUST be as defined in section 2.2.15 Vulnerability Involvement Type Model. » [CSAF-6.7.1-3]

6.7.1.1 Vulnerability – Involvements – Involvement – Description

Element vuln:Description

« The vuln:Description element MUST occur zero or one time in vuln:Involvment and if present contains a string value, that represents a thorough human-readable discussion of the **Involvement**. » [CSAF-6.7.1.1-1]

Example 53:

```
<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 54:

```
<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>
```

Example 55:

```
<Involvements>
  <Involvement Party="Vendor" Status="Completed"></Involvement>
</Involvements>
```

6.8 Vulnerability – CVE

Element `vuln:CVE`

« The `vuln:CVE` element MUST be present zero or one time in any `vuln:Vulnerability` and if present its value holds the MITRE standard Common Vulnerabilities and Exposures (CVE) tracking number for the vulnerability and this value MUST match the pattern documented in section 2.2.10 Vulnerability CVE Type Model. » [CSAF-6.8-1]

Non-normative comment:

CVE is a standard for vulnerability naming that provides improved tracking of vulnerabilities over time across different reporting sources. More information about CVE domain values can be found in section 2.2.10 Vulnerability CVE Type Model.

Example 56:

```
<CVE>CVE-2010-3864</CVE>
```

6.9 Vulnerability – CWE

Element `vuln:CWE`

« The `vuln:CWE` element MUST be present zero or one time in any `vuln:Vulnerability` and if present it contains the MITRE standard Common Weakness Enumeration (CWE) and this value MUST match the pattern documented in section 2.2.13 Vulnerability CWE Type Model. » [CSAF-6.9-1]

Non-normative comment:

The CWE domain value type is described in section 2.2.13 Vulnerability CWE Type Model.

Example 57:

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

Example 58:

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

6.10 Vulnerability – Product Statuses

Element `vuln:ProductStatuses`

« The `vuln:ProductStatuses` element MUST be present zero or one time in any `vuln:Vulnerability` and if present it MUST contain one or more `vuln:Status` elements which contain a subset of products chosen from **Product Tree**. » [CSAF-6.10-1]

Every affected (and unaffected) product relating to the vulnerability is referenced here, inside one or more **Status** containers.

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.

Non-normative comment:

Without this constraint, it would be possible to assign conflicting status information to one and the same product.

6.10.1 Vulnerability – Product Statuses – Status

Element `vuln:Status`

« The `vuln:Status` element MUST be present with cardinality [1, ∞] in `vuln:ProductStatuses` and MUST contain one or more `vuln:ProductID` elements. » [CSAF-6.10.1-1]

Attribute Type

The allowed values of the `Type` attribute of a `vuln:Status` element are documented in section 2.2.16 Vulnerability Product Affected Status Type Model. 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.

6.10.1.1 Vulnerability – Product Statuses – Status – Product ID

Element `vuln:ProductID`

« The `vuln:ProductID` element MUST be present one or more times inside a `vuln:Status` element and defines a product as having the status defined in the parent element's `Type` attribute. » [CSAF-6.10.1.1-1]

The reference is made via value by using the unique `ProductID` attribute of a **Full Product Name** element that is defined in the **Product Tree**.

« A single **Product ID** MUST not be assigned more than one status type within the same **Vulnerability**. » [CSAF-6.10.1.1-2]

Example 59:

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

Element `vuln:Threats`

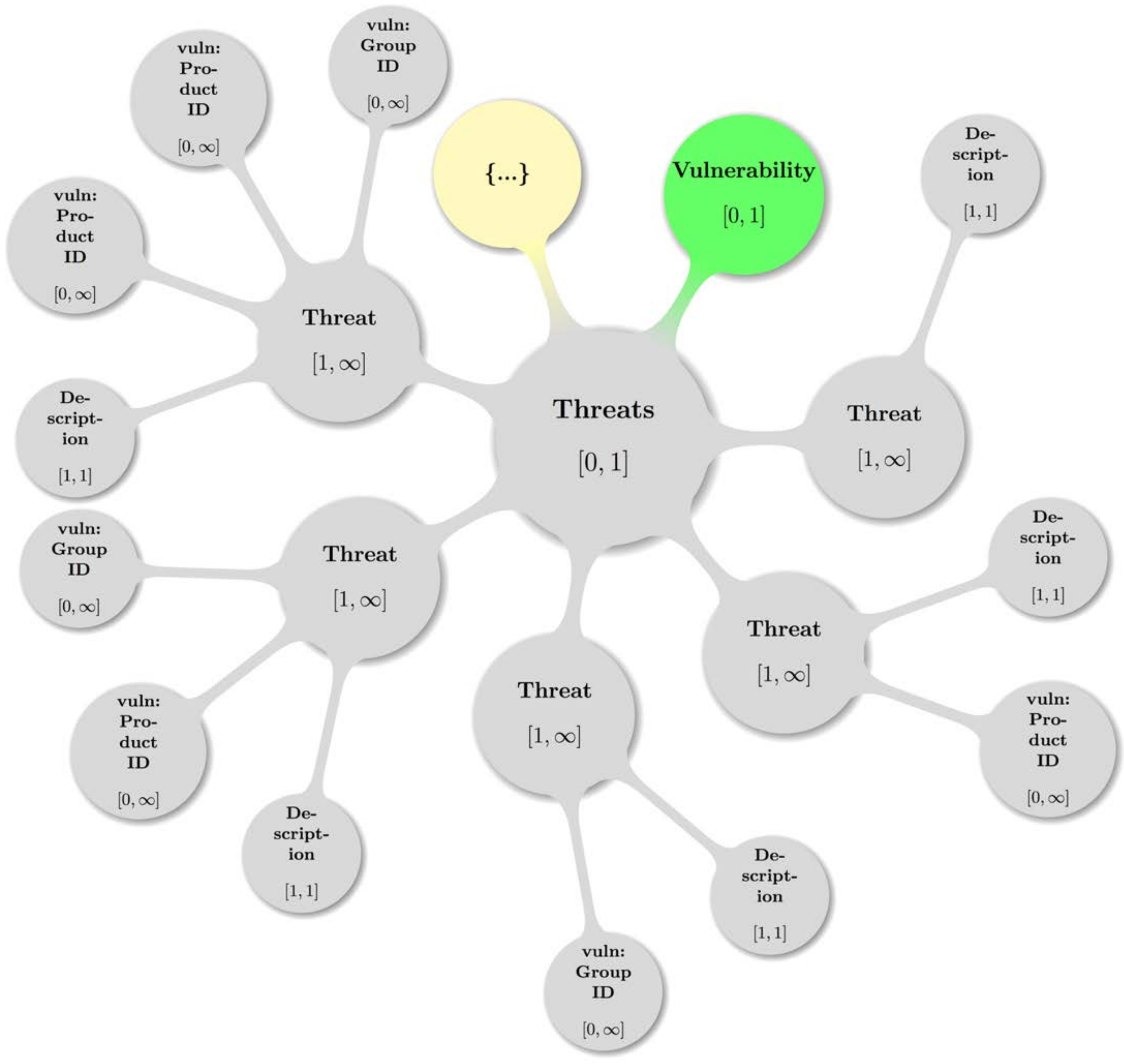
« The `vuln:Threats` element MUST be present zero or one time inside any `vuln:Vulnerability` and if present it MUST contain one or more `vuln:Threat` elements which contain information about a vulnerability that can change with time. » [CSAF-6.11-1]

Non-normative comment:

Threat information about a vulnerability that can change with time is also called "vulnerability kinetics".

A visual map of some valid **Threats** configuration including the parent node (**Vulnerability**) — again with the node labeled {…} indicating further possible **Threat** subtrees follows below.

*Figure 11: A topologically valid **Threats** configuration.*



6.11.1 Vulnerability – Threats – Threat

Element `vuln:Threat`

« The `vuln:Threat` element MUST be present with cardinality [1, ∞] as in `vuln:Threats` and MUST contain exactly one `vuln:Description` element and zero or more `vuln:ProductID` and `vuln:GroupID` elements and in that order. » [CSAF-6.11.1-1]

The element `vuln:Threat` contains the vulnerability kinetic information. This information can change as the vulnerability ages and new information becomes available.

The attribute `Type` is required and categorizes the threat according to the rules in section 2.2.18 Vulnerability Threat Type Model.

A **Threat** container is 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.

Attribute Type

The allowed enumerated values for the `Type` attribute are documented in section 2.2.18 Vulnerability Threat Type Model.

Attribute `Date`

The `Date` attribute is optional, but if given it must contain a date time value as documented in section 2.2.1 Date and Time.

6.11.1.1 Vulnerability – Threats – Threat – Description

Element `vuln:Description`

« The `vuln:Description` element MUST be present exactly once in `vuln:Threat` and the string content represents a thorough human-readable discussion of the **Threat**. » [CSAF-6.11.1.1-1]

Example 60: Impact:

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

Example 61: Exploit Status:

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

Example 62: 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 63: Target Set:

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

Example 64: Target Set variation of content:

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

Example 65: 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

Element `vuln:ProductID`

« The `vuln:ProductID` element MUST be present with the cardinality [0, ∞] in `vuln:Threat` element and if given represents a reference by value to the related product via the unique `ProductID` attribute of the matching **Full Product Name** element. » [CSAF-6.11.1.2-1]

If a **Threat** applies to more than one Product, either additional **Product ID** elements or **Group ID** elements (replacing/combining those) SHOULD be added.

6.11.1.3 Vulnerability – Threats – Threat – Group ID

Element `vuln:GroupID`

« The `vuln:GroupID` element MUST be present with the cardinality [0, ∞] in a `vuln:Threat` element and if given represents a reference by value to the related products via the unique `GroupID` attribute of a **Group** element that is defined in the **Product Tree**. » [CSAF-6.11.1.3-1]

If the **Threat** pertains to several products that have been logically grouped into a **Product Group** optional element `vuln:GroupID` represents a reference to that group of products.

If a **Threat** applies to more than one group of products, multiple **Group ID** elements are added accordingly.

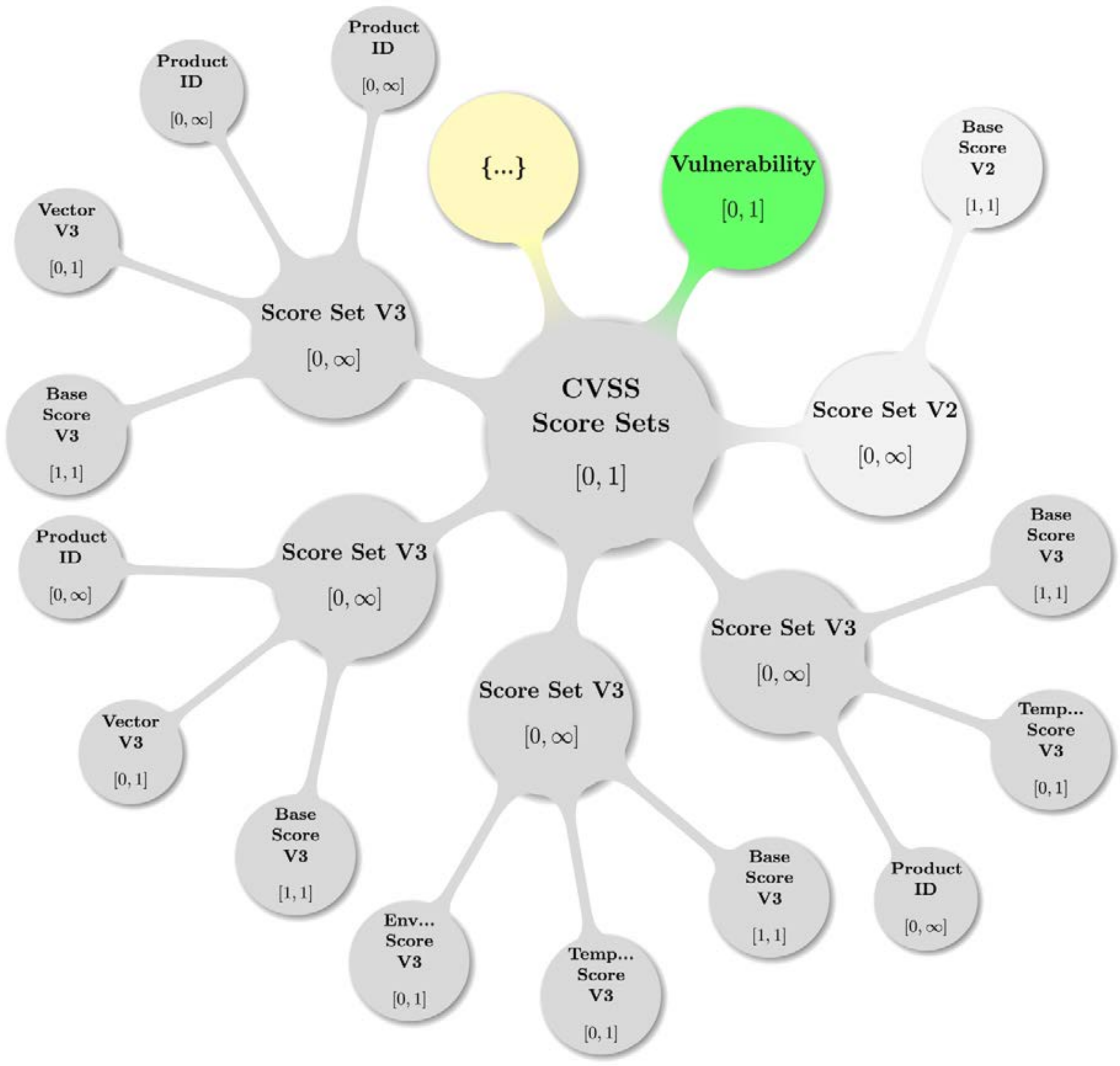
6.12 Vulnerability – CVSS Score Sets

Element `vuln:CVSSScoreSets`

« The `vuln:CVSSScoreSets` element MUST present zero or one time per and inside any `vuln:Vulnerability` and holds one or more of the `vuln:ScoreSetV2` (deprecated) or `vuln:ScoreSetV3` (preferred) container elements and in that order if instances of both are present. » [CSAF-6.12-1]

A visual map of some valid **CVSS Score Sets** configuration including the parent node (**Vulnerability**) — again the node with label {…} indicates further possible **Score Set V3** (preferred) or ***Score Set V2*** (deprecated) subtrees is shown below:

*Figure 12 A topologically valid **CVSS Score Sets** configuration.*



6.12.1 Vulnerability – CVSS Score Sets – Score Set V2

Element `vuln:ScoreSetV2`

« The `vuln:ScoreSetV2` element MUST be present with cardinality [0, ∞] inside `vuln:CVSSScoreSets` and if given, every instance MUST hold at least exactly one `vuln:BaseScoreV2` element. » [CSAF-6.12.1-1]

« Any `vuln:ScoreSetV2` instance MAY additionally provide the further children: `vuln:TemporalScoreV2` [0, 1], `vuln:EnvironmentalScoreV2` [0, 1], `vuln:VectorV2` [0, 1], `vuln:ProductID` [0, ∞] and in that order. » [CSAF-6.12.1-2]

If a value of the temporal or environmental score is set to "not defined", the corresponding elements SHOULD be omitted.

The allowed values for the children of the `vuln:ScoreSetV2` element are documented in section 2.2.11 Vulnerability CVSS Version 2 Type Model.

Non-normative comment:

If given, instances hold the actual CVSS version 2 metrics [CVSS2]

See also section 2.2.11 Vulnerability CVSS Version 2 Type Model for further information and constraints on the values of the containers components.

A **Score Set V2** container can be tied to one or more specific products by referencing these products using the **Product ID** child element. If the **Score Set V2** 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 V2** 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

Element `vuln:BaseScoreV2`

« The `vuln:BaseScoreV2` element MUST be present exactly once inside every `vuln:ScoreSetV2` and contains the numeric value of the computed CVSS version 2 base score.
» [CSAF-6.12.1.1-1]

The finite set of allowed values the `vuln:BaseScoreV2` element is documented in section 2.2.11 Vulnerability CVSS Version 2 Type Model.

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

Element `vuln:TemporalScoreV2`

« The `vuln:TemporalScoreV2` element MUST be present zero or one time inside any `vuln:ScoreSetV2` and if given contains the numeric value of the computed CVSS version 2 temporal score. » [CSAF-6.12.1.2-1]

The finite set of allowed values for the `vuln:TemporalScoreV2` element is documented in section 2.2.11 Vulnerability CVSS Version 2 Type Model.

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

Element `vuln:EnvironmentalScoreV2`

« The `vuln:EnvironmentalScoreV2` element MUST be present zero or one time inside any `vuln:ScoreSetV2` and if given contains the numeric value of the computed CVSS version 2 environmental score. » [CSAF-6.12.1.3-1]

The finite set of allowed values for the `vuln:EnvironmentalScoreV2` element are documented in section 2.2.11 Vulnerability CVSS Version 2 Type Model.

Non-normative comment:

This metric is typically reserved for use by the end user and is specific to the environment in which the affected product is deployed. See also section 2.2.11 Vulnerability CVSS Version 2 Type Model for further information and constraints on this element.

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

Element `vuln:VectorV2`

« The `vuln:VectorV2` element MUST be present zero or one time inside any `vuln:ScoreSetV2` and if present contains the official string notation that displays all the values used to compute the CVSS version 2 base, temporal, and environmental scores. » [CSAF-6.12.1.4-1]

The allowed values for the `vuln:VectorV3` element are documented in section 2.2.11 Vulnerability CVSS Version 2 Type Model.

Example 66:

```
<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

Element `vuln:ProductID`

« The `vuln:ProductID` element MUST be present with cardinality [0, ∞| inside any `vuln:ScoreSetV2` element and per instance value references to unique `ProductID` attributes of **Full Product Name** elements defined in the **Product Tree** are noted. » [CSAF-6.12.1.5-1]

If a **Score Set V2** applies to more than one product, you can add multiple **Product ID** elements as references accordingly.

Any single **Product ID** is to be assigned to exactly none or one **Score Set V2** within the same **Vulnerability**.

6.12.2 Vulnerability – CVSS Score Sets – Score Set V3

Element `vuln:ScoreSetV3`

« The `vuln:ScoreSetV3` element MUST be present with cardinality [0, ∞] inside `vuln:CVSSScoreSets` and if given, every instance MUST hold at least exactly one `vuln:BaseScoreV3` element. » [CSAF-6.12.2-1]

« Any `vuln:ScoreSetV3` instance MAY additionally provide the further children: `vuln:TemporalScoreV3` [0, 1], `vuln:EnvironmentalScoreV3` [0, 1], `vuln:VectorV3` [0, 1], `vuln:ProductID` [0, ∞] and in that order. » [CSAF-6.12.2-2]

If a value of the temporal or environmental score is set to "not defined", the corresponding elements SHOULD be omitted.

The allowed values for the children of the `vuln:ScoreSetV3` element are documented in section 2.2.12 Vulnerability CVSS Version 3 Type Model.

Non-normative comment:

If given, instances hold the actual CVSS version 3 metrics [CVSS3].

See also section 2.2.12 Vulnerability CVSS Version 3 Type Model for further information and constraints on the values of the containers components.

A **Score Set V3** container can be tied to one or more specific products by referencing these products using the **Product ID** child element. If the **Score Set V3** 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 V2** 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

Element `vuln:BaseScoreV3`

« The `vuln:BaseScoreV3` element MUST be present exactly once inside every `vuln:ScoreSetV3` and contains the numeric value of the computed CVSS version 3 base score. » [CSAF-6.12.2.1-1]

The finite set of allowed values for the `vuln:BaseScoreV3` element is documented in section 2.2.12 Vulnerability CVSS Version 3 Type Model.

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

Element `vuln:TemporalScoreV3`

« The `vuln:TemporalScoreV3` element MUST be present zero or one time inside any `vuln:ScoreSetV3` and if given contains the numeric value of the computed CVSS version 3 temporal score. » [CSAF-6.12.2.2-1]

The finite set of allowed values for the `vuln:TemporalScoreV3` element is documented in section 2.2.12 Vulnerability CVSS Version 3 Type Model.

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

Element `vuln:EnvironmentalScoreV3`

« The `vuln:EnvironmentalScoreV3` element MUST be present zero or one time inside any `vuln:ScoreSetV3` and if given contains the numeric value of the computed CVSS version 3 environmental score. » [CSAF-6.12.2.3-1]

The finite set of allowed values for the `vuln:EnvironmentalScoreV3` element are documented in section 2.2.12 Vulnerability CVSS Version 3 Type Model.

Non-normative comment:

This metric is typically reserved for use by the end user and is specific to the environment in which the affected product is deployed. See also section 2.2.12 Vulnerability CVSS Version 3 Type Model for further information and constraints on this element.

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

Element `vuln:VectorV3`

« The `vuln:VectorV3` element MUST be present zero or one time inside any `vuln:ScoreSetV3` and if present contains the official string notation that displays all the values used to compute the CVSS version 3 base, temporal, and environmental scores. » [CSAF-6.12.2.4-1]

The allowed values for the `vuln:VectorV3` element are documented in section 2.2.12 Vulnerability CVSS Version 3 Type Model.

Example 67:

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

Example 68:

```
<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

Element `vuln:ProductID`

« The `vuln:ProductID` element MUST be present with cardinality [0, ∞| inside any `vuln:ScoreSetV3` element and per instance value references to unique `ProductID` attributes of **Full Product Name** elements defined in the **Product Tree** are noted.
» [CSAF-6.12.2.5-1]

If a **Score Set V3** applies to more than one product, you can add multiple **Product ID** elements as references accordingly.

Any single **Product ID** is to be assigned to exactly none or one **Score Set V3** within the same **Vulnerability**.

6.13 Vulnerability – Remediations

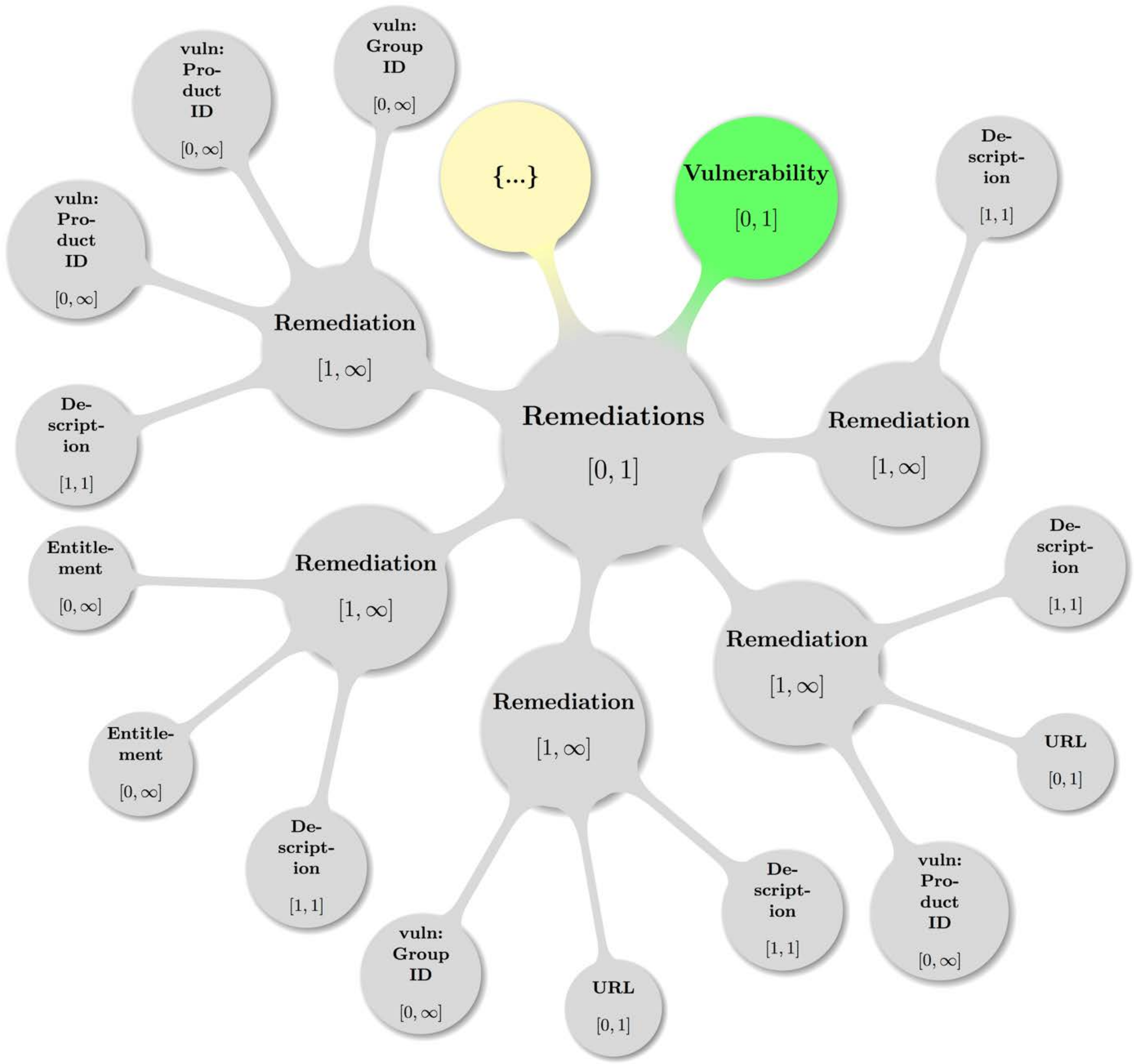
Element `vuln:Remediations`

« The `vuln:Remediations` element MUST be present with cardinality [0, 1] inside `vuln:Vulnerability` and it holds [1, ∞] `vuln:Remediation` child elements.
» [CSAF-6.13-1]

These **Remediation** containers will have details on how to remediate a vulnerability.

Visual display of a map of some valid **Remediations** configuration including the parent node (**Vulnerability**) — again with the node labeled {…} indicating further possible **Remediation** subtrees — follows below.

*Figure 13: A topologically valid **Remediations** configuration.*



6.13.1 Vulnerability – Remediations – Remediation

Element `vuln:Remediation`

« The `vuln:Remediation` element MUST be present with cardinality [1, ∞] in `vuln:Remediations` and MUST contain the following child elements `vuln:Description` [1, 1], `vuln:Entitlement` [0, ∞], `vuln:URL` [0, 1], `vuln:ProductID` [0, ∞], and `vuln:GroupID` [0, ∞] in that order. » [CSAF-6.13.1-1]

The element `vuln:Remediation` is a container that holds specific details on how to handle (and presumably, fix) a vulnerability.

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.

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

Attribute Type

The allowed values for the required `Type` attribute are documented in section 2.2.17 Vulnerability Remediation Type Model.

6.13.1.1 Vulnerability – Remediations – Remediation – Description

Element `vuln:Description`

« The `vuln:Description` element MUST be present exactly once in `vuln:Remediation` and it contains a thorough human-readable discussion of the **Remediation**. » [CSAF-6.13.1.1-1]

6.13.1.2 Vulnerability – Remediations – Remediation – Entitlement

Element `vuln:Entitlement`

« The `vuln:Entitlement` element MUST be present with cardinality [0, ∞] inside `vuln:Remediation` and it contains any possible vendor-defined constraints for obtaining fixed software or hardware that fully resolves the vulnerability. » [CSAF-6.13.1.2-1]

Non-normative comment:

This element will often contain information about service contracts or service-level agreements that is directed toward customers of large vendors.

Example 69:

```
<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

Element `vuln:URL`

« The `vuln:URL` element MUST be present with cardinality [0, 1] in `vuln:Remediation` and it contains the URL to the **Remediation**. » [CSAF-6.13.1.3-1]

6.13.1.4 Vulnerability – Remediations – Remediation – Product ID

Element `vuln:ProductID`

« The `vuln:ProductID` element MUST be present with cardinality [0, ∞] inside `vuln:Remediation` and the content of the instances are tokens that are references to products through the value of a **Full Product Name**'s `ProductID` attribute.
» [CSAF-6.13.1.4-1]

If the **Remediation** pertains to a specific product, a `vuln:ProductID` represents the reference to that product.

The reference is made using the unique `ProductID` attribute of a **Full Product Name** element that is defined in the **Product Tree**.

If a **Remediation** applies to more than one Product, multiple **Product ID** elements SHOULD be added accordingly, or the **Group ID** element (see below) instead.

Example 70:

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

6.13.1.5 Vulnerability – Remediations – Remediation – Group ID

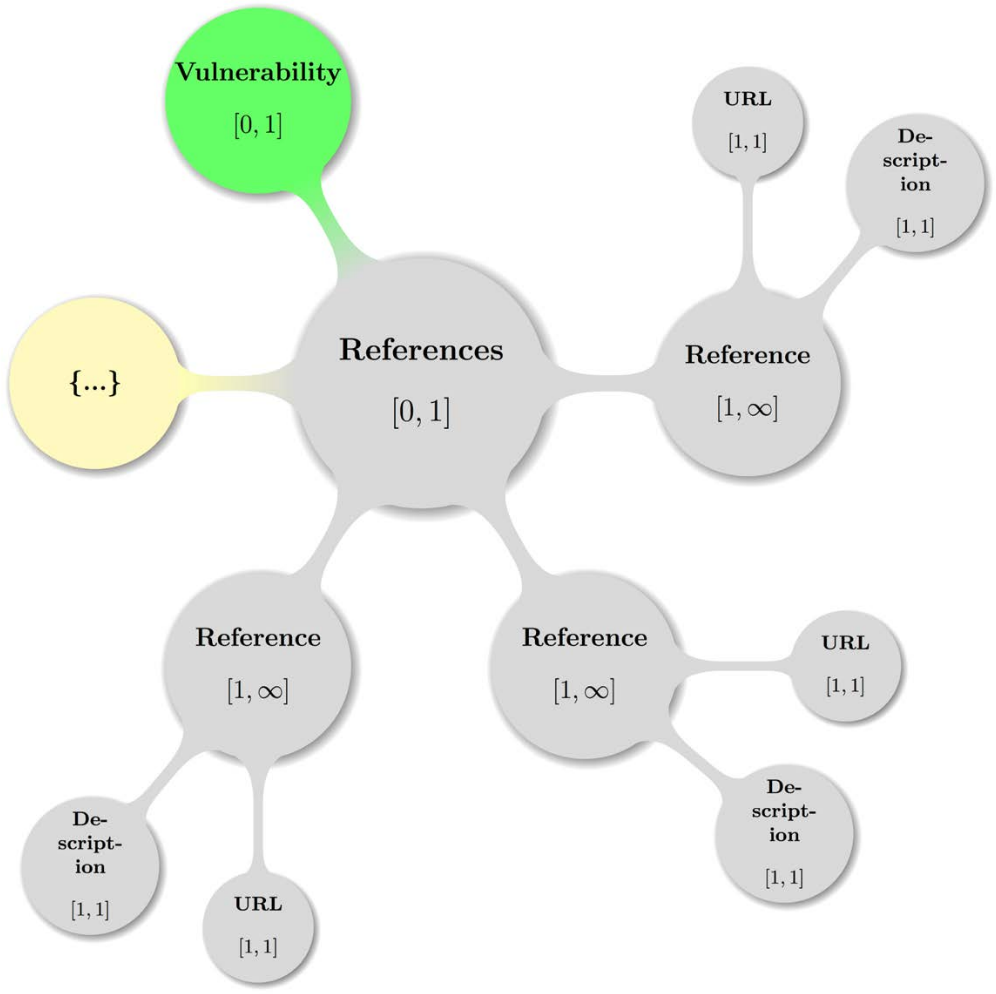
> **Element `vuln:GroupID`**
>
> « The `vuln:GroupID` element MUST be present with cardinality [0, ∞] inside `vuln:Remediation` and the content of the instances are tokens that are references to groups of products. » [CSAF-6.13.1.5-1]
>
> If the **Remediation** pertains to several products that have 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 `GroupID` attribute of a **Group** element that is defined in the **Product Tree**.
>
> If a **Remediation** applies to more than one group of products, one can add multiple **Group ID** elements accordingly.

6.14 Vulnerability – References

> **Element `vuln:References`**
>
> « The `vuln:References` element MUST be present with cardinality [0, 1] inside `vuln:Vulnerability` parent at last position or before any Acknowledgements if these exist. » [CSAF-6.14-1]
>
> 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.
>
> « If present, the `vuln:References` MUST contain [1, ∞] `vuln:Reference` child element instances. » [CSAF-6.14-2]

A visual map of some valid **References** configuration including the parent node (**Vulnerability**) — again with the node labeled {…} indicating further possible **Reference** subtrees is provided below:

*Figure 14: A topologically valid **Vulnerability References** configuration.*



6.14.1 Vulnerability – References – Reference

Element `vuln:Reference`

« The `vuln:Reference` element MUST be present with cardinality [1, ∞] inside `vuln:References` and every instance MUST have exactly one `vuln:URL` and one `vuln:Description` child element and in that order. » [CSAF-6.14.1-1]

The element `vuln:Reference` contains a description of a related document specific to a vulnerability section of a CVRF document.

Attribute Type

« The Attribute `Type` if not present is taken to be the enumeration value `External` and if present MUST be one of the values documented in section 2.2.7 Reference Type Model. » [CSAF-6.14.1-2]

This attributes value denotes the type of the document reference relative to the CSAF CVRF document itself

Non-normative comment:

This may include a plaintext or HTML version of the advisory or other related documentation, such as white papers or mitigation documentation.

6.14.1.1 Vulnerability – References – Reference – URL

Element `vuln:URL`

« The `vuln:URL` element MUST be present exactly once in `vuln:Reference` and contains the fixed URL or location of the reference. » [CSAF-6.14.1.1-1]

6.14.1.2 Vulnerability – References – Reference – Description

Element `vuln:Description`

« The `vuln:Description` MUST be present exactly once in `vuln:Reference` and holds a descriptive title or name of the reference. » [CSAF-6.14.1.2-1]

Example 71:

```
<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

Element `vuln:Acknowledgements`

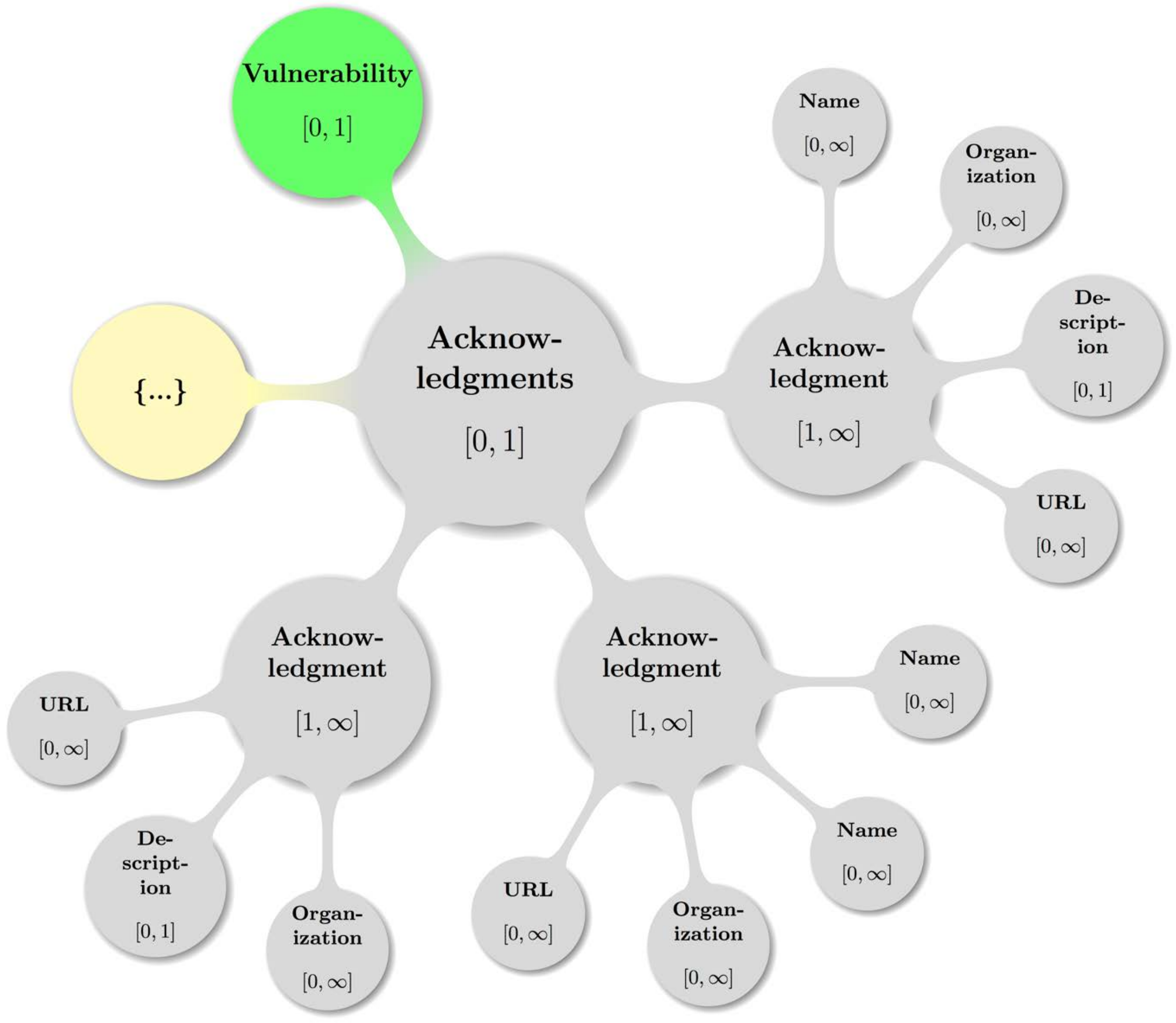
« The `vuln:Acknowledgements` element MUST be present with cardinality [0, 1] inside `vuln:Vulnerability` element and MUST contain one or more `vuln:Acknowledgement` child elements, which in turn either contain(s) recognition of external parties or is empty. » [CSAF-6.15-1]

Non-normative comment:

This **Acknowledgments** container is different from the one at the document level because it is specifically related to the vulnerability in context**.**

Following is a map of a valid **Acknowledgements** configuration including the parent node (**Vulnerability**) — again with the node labeled {…} indicating further possible **Acknowledgement** subtrees:

*Figure 15: A topologically valid **Vulnerability Acknowledgements** configuration.*



6.15.1 Vulnerability – Acknowledgements – Acknowledgement

Element `vuln:Acknowledgement`

« The `vuln:Acknowledgement` element MUST be present with cardinality [1, ∞] inside `vuln:Acknowledgments` and MUST contain the following child elements `vuln:Name` [0, ∞] times, `vuln:Organization` [0, ∞] times, `vuln:Description` [0, 1] times, and `vuln:URL` [0, ∞] times and in that order. » [CSAF-6.15.1-1]

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.

Non-normative comment:

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.

6.15.1.1 Vulnerability – Acknowledgements – Acknowledgement – Name

Element `vuln:Name`

« The `vuln:Name` MUST be present with cardinality [1, ∞] in `vuln:Acknowledgment` and every instance given contains the name of the party being acknowledged. » [CSAF-6.15.1.1-1]

6.15.1.2 Vulnerability – Acknowledgements – Acknowledgement – Organization

Element `vuln:Organization`

« The `vuln:Organization` MUST be present with cardinality [0, ∞] inside `vuln:Acknowledgment`. » [CSAF-6.15.1.2-1]

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

6.15.1.3 Vulnerability – Acknowledgements – Acknowledgement – Description

Element `vuln:Description`

« The `vuln:Description` element MUST be present with cardinality [0, 1] inside `vuln:Acknowledgment` and if given contains any contextual details the document producers wish to make known about the acknowledgment or acknowledged parties. » [CSAF-6.15.1.3-1]

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.

6.15.1.4 Vulnerability – Acknowledgements – Acknowledgement – URL

Element `vuln:URL`

« The `vuln:URL` MUST be present with cardinality [0, ∞] inside `vuln:Acknowledgment` and every instance contains the URL or location of the reference to be acknowledged.
» [CSAF-6.15.1.4-1]

Example 72:Taking Isaac Newton, Jeanne D'Arc, and Alan Turing as names for fictitious people, and Acme, as well as Things as such organizations and some vendor Vendorix:

```
<Acknowledgments>
  <Acknowledgment>
    <Name>Isaac Newton</Name>
    <Name>Jeanne D'Arc</Name>
    <Organization>Acme</Organization>
    <URL>http://acme.example.com/IsaacAndJeanne/</URL>
  </Acknowledgment>
  <Acknowledgment>
    <Name>Alan Turing</Name>
    <Organization>Things</Organization>
    <Description>
      Vendorix would like to thank Alan Turing from Things for reporting this issue.
    </Description>
    <URL>http://things.example.com/JustAlan/</URL>
  </Acknowledgment>
  <Acknowledgment>
    <Description>
      Vendorix would like to thank the following researchers for their contributions to
      making this project more secure: Isaac Newton, Jeanne D'Arc, Alan Turing
    </Description>
    <URL>http://white-hats.example.com/AllOfThemAgain/</URL>
  </Acknowledgment>
</Acknowledgments>
```

7 Conformance

7.1 Conformance as a CSAF CVRF version 1.2 document

To ease communication and subsequent resolution of any specific partial conformance violation, the preceding chapters already provide minimal requirements, that a specific instance component must fulfill, to permit conformance of the complete CSAF CVRF version 1.2 document.

The following clause offers a simple seven step process, to either prove or disprove the conformance of a complete XML document (formulated in terms specific to that implementation language) to this version of CSAF CVRF:

« A document instance conforms to this specification as a CSAF CVRF document if it meets all of 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.

» [CSAF-7.1-1] **and** the date and version values fulfill the following four additional requirements:

4. Requirement CSAF-2.2.1-1: "Adherence to ISO 8601 [ISO8601]" (cf. 2.2.1 Date and Time Model)
5. Requirement CSAF-2.2.1-2: "Offset to UTC convention" (cf. 2.2.1 Date and Time Model)
6. Requirement CSAF-2.2.1-3: "T Separator of Date and Time" (cf. 2.2.1 Date and Time Model)
7. Requirement CSAF-2.2.9-1: "Version field increment convention" (cf. 2.2.9 Version Type Model)

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
Ben Sooter, Electric Power Research Institute
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 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, McAfee
Lothar Braun, Siemens AG
Louis Ronnau, Cisco Systems
Mark Davidson, NC4
Mark-David McLaughlin, Cisco Systems
Masato Terada, "Hitachi, Ltd."
Masood Nasir, TELUS
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
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. Table of XML Elements and Attributes

Appendix C. List of Figures

Appendix D. Demo of a Document Schema Migration

To estimate the efforts required for a migration from a CVRF version 1.1 to a CSAF CVRF version 1.2 document, members of the committee have transformed a large body of existing CVRF v1.1 documents into the new format.

Some typical procedures and real world diffs (edited for brevity) are given in this appendix and as service to the reader.

D.1 Transform A — Producer transports CVSSv3 in v1.1 ScoreSet

As many vendors already started scoring the vulnerabilities under the CVSS v3 ruling, there was a need to transport these values in the CVRF v1.1 context. It is common practice of producers thus to announce “out of band” somewhere and prominently that the scores communicated are CVSS version 3 scores and then expect the consumers to interpret the content of CVRF version 1.2 document’s ScoreSet elements to be interpreted this way.

A minor complication could be, that a) CVSS version 3 vectors are expected to have the constant version prefix (which is sometimes missing in these documents) and that b) it is possible, that a version3 CVSS vector does not fit inside the CVRF 1.1 length limit for the Vector element.

Care has been taken during the design phase of CSAF CVRF version 1.2 to add both CVSS score set “universes” (v2 and v3) side by side and renaming the v2 variant from CVRF 1.1 into ScoreSetV2 with the children BaseScoreV2, EnvironmentalScoreV2, TemporalScoreV2 and VectorV2 — and correspondingly with the a V3 postfix the CVSSv3 elements.

D.1.1 The actual CVSS Score Set V2/3 transform procedure

For every document (if and only if the values represent CVSSv3 scores!):

1. Map the `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSet`
   into `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3`
2. Map the `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/BaseScore`
   into `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/BaseScoreV3`
3. Map any `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/EnvironmentalScore`
   into `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/EnvironmentalScoreV3`
4. Map any `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/TemporalScore`
   into `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/TemporalScoreV3`
5. Map any `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/Vector`
   into `/cvrfdoc/Vulnerability/CVSSScoreSets/ScoreSetV3/VectorV3`
6. Transform any non-prefixed VectorV3 content into the correct CVSSv3 form.

D.2 Transform B — Empty v1.1 Acknowledgement/Organization

In CVRF v1.1 documents, it is allowed – albeit redundant – to have an empty Organization element inside of an Acknowledgement element (instead of simply leaving it out).

In CSAF CVRF v1.2 an empty Organization element inside of an Acknowledgement element is invalid.

D.2.1 The actual removal transform procedure

For every document:

1. Remove any occurrence of empty `/cvrfdoc/Acknowledgements/Acknowledgement/Organization.`

Appendix E. Complete 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).

E.1 Sample Security Advisory A

Security advisory from the year 2017:

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

E.2 Sample Security Advisory B

Another security advisory from the year 2017 as issued by Red Hat and migrated to CSAF CVRF 1.2 from the source CVRF 1.1 by simply updating the namespaces and prefixing all elements with the corresponding namespace of either `cvrf`, `prod`, or `vuln.` Additionally, 4 comments were added, to visually separate the three semantic top level elements **Document** Context, **Product Tree**, and **Vulnerability**):

Example 73:

```
<?xml version="1.0" encoding="utf-8"?>
<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/1.1/"
  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
  xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf"
  >
  <!-- Document wide context information -->
  <DocumentTitle xml:lang="en">Red Hat Security Advisory: python-oslo-middleware
security update</DocumentTitle>
  <DocumentType>Security Advisory</DocumentType>
  <DocumentPublisher Type="Vendor">
    <ContactDetails>secalert@redhat.com</ContactDetails>
    <IssuingAuthority>Red Hat Product Security</IssuingAuthority>
  </DocumentPublisher>
  <DocumentTracking>
    <Identification>
      <ID>RHSA-2017:0435</ID>
    </Identification>
    <Status>Final</Status>
    <Version>1</Version>
    <RevisionHistory>
      <Revision>
        <Number>1</Number>
        <Date>2017-03-02T21:13:00Z</Date>
        <Description>Current version</Description>
      </Revision>
    </RevisionHistory>
    <InitialReleaseDate>2017-03-02T21:13:00Z</InitialReleaseDate>
    <CurrentReleaseDate>2017-03-02T21:13:00Z</CurrentReleaseDate>
    <Generator>
      <Engine>Red Hat rhsa-to-cvrf 2.0</Engine>
      <Date>2017-03-04T05:06:05Z</Date>
    </Generator>
  </DocumentTracking>
  <DocumentNotes>
    <Note Title="Topic" Type="Summary" Ordinal="1" xml:lang="en">
An update for python-oslo-middleware is now available for Red Hat OpenStack Platform 9.0
(Mitaka).

Red Hat Product Security has rated this update as having a security impact of Moderate.
A Common Vulnerability Scoring System (CVSS) base score, which gives a detailed severity
rating, is available for each vulnerability from the CVE link(s) in the References
section.    </Note>
    <Note Title="Details" Type="General" Ordinal="2" xml:lang="en">
The OpenStack Oslo Middleware library provides components that can be injected into WSGI
pipelines to intercept request and response flows. The base class can be enhanced with
functionality like adding or updating HTTP headers, or to offer support for limiting
size or connections.

Security Fix(es):

* An information-disclosure flaw was found in oslo.middleware. Software using the
CatchError class could include sensitive values in a traceback's error message. System
users could exploit this flaw to obtain sensitive information from OpenStack component
error logs (for example, keystone tokens). (CVE-2017-2592)
```

```
Red Hat would like to thank the OpenStack project for reporting this issue. Upstream
acknowledges Divya K Konoor (IBM) as the original reporter.    </Note>
    <Note Title="Terms of Use" Ordinal="3" Type="Legal Disclaimer" xml:lang="en">Please
see https://www.redhat.com/footer/terms-of-use.html</Note>
  </DocumentNotes>
  <DocumentDistribution xml:lang="en">Copyright © 2017 Red Hat, Inc. All rights
reserved.</DocumentDistribution>
  <AggregateSeverity
Namespace="https://access.redhat.com/security/updates/classification/">Moderate</Aggrega
teSeverity>
  <DocumentReferences>
    <Reference Type="Self">
      <URL>https://rhn.redhat.com/errata/RHSA-2017-0435.html</URL>
      <Description>https://rhn.redhat.com/errata/RHSA-2017-0435.html</Description>
    </Reference>
    <Reference>
      <URL>https://access.redhat.com/security/updates/classification/#moderate</URL>

<Description>https://access.redhat.com/security/updates/classification/#moderate</Descri
ption>
    </Reference>
  </DocumentReferences>
  <!-- Product tree section -->
  <prod:ProductTree xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/prod">
    <Branch Type="Product Family" Name="Red Hat Enterprise Linux OpenStack Platform">
      <Branch Type="Product Name" Name="Red Hat OpenStack Platform 9.0">
        <FullProductName ProductID="7Server-RH7-RHOS-9.0">Red Hat OpenStack Platform
9.0</FullProductName>
      </Branch>
    </Branch>
    <Branch Type="Product Version" Name="python-oslo-middleware-3.7.0-2.el7ost">
      <FullProductName ProductID="python-oslo-middleware-3.7.0-2.el7ost">python-oslo-
middleware-3.7.0-2.el7ost.src.rpm</prod:FullProductName>
    </Branch>
    <Relationship ProductReference="python-oslo-middleware-3.7.0-2.el7ost"
      RelationType="Default Component Of" RelatesToProductReference="7Server-RH7-RHOS-
9.0">
      <FullProductName ProductID="7Server-RH7-RHOS-9.0:python-oslo-middleware-3.7.0-
2.el7ost">python-oslo-middleware-3.7.0-2.el7ost as a component of Red Hat OpenStack
Platform 9.0</FullProductName>
    </Relationship>
  </ProductTree>
  <!-- Vulnerability section -->
  <vuln:Vulnerability Ordinal="1"
   xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/vuln">
    <Notes>
      <Note Title="Vulnerability Description" Type="General" Ordinal="1"
xml:lang="en">An information-disclosure flaw was found in oslo.middleware. Software
using the CatchError class could include sensitive values in a traceback's error
message. System users could exploit this flaw to obtain sensitive information from
OpenStack component error logs (for example, keystone tokens). </Note>
    </Notes>
    <DiscoveryDate>2017-01-18T00:00:00Z</DiscoveryDate>
    <ReleaseDate>2017-01-26T00:00:00Z</ReleaseDate>
    <Involvements>
      <Involvement Party="Vendor" Status="Completed"/>
    </Involvements>
    <CVE>CVE-2017-2592</CVE>
    <ProductStatuses>
      <Status Type="Fixed">
        <ProductID>7Server-RH7-RHOS-9.0:python-oslo-middleware-3.7.0-
2.el7ost</ProductID>
      </Status>
    </ProductStatuses>
    <Threats>
      <Threat Type="Impact">
        <Description>Moderate</Description>
      </Threat>
    </Threats>
```

```xml
    <Remediations>
      <Remediation Type="Vendor Fix">
        <Description xml:lang="en">
For details on how to apply this update, which includes the changes described in this
advisory, refer to:

https://access.redhat.com/articles/11258    </Description>
        <URL>https://rhn.redhat.com/errata/RHSA-2017-0435.html</URL>
      </Remediation>
    </Remediations>
    <References>
      <Reference>
        <URL>https://access.redhat.com/security/cve/CVE-2017-2592</URL>
        <Description>CVE-2017-2592</vuln:Description>
      </Reference>
      <Reference>
        <URL>https://bugzilla.redhat.com/show_bug.cgi?id=1414698</URL>
        <Description>bz#1414698: CVE-2017-2592 python-oslo-middleware: CatchErrors leaks
sensitive values into error logs</Description>
      </Reference>
    </References>
    <Acknowledgments>
      <Acknowledgment>
        <Description>Red Hat would like to thank the OpenStack project for reporting
this issue. Upstream acknowledges Divya K Konoor (IBM) as the original
reporter.</Description>
      </Acknowledgment>
    </Acknowledgments>
  </Vulnerability>
  <!-- No more elements to follow -->
</cvrfdoc>
```

E.3 Sample Security Advisory C

Yet another security advisory from the year 2017 as issued by Cisco and migrated to CSAF CVRF 1.2 from the source CVRF 1.1 by simply updating the namespaces and prefixing all elements with the corresponding namespace of either `cvrf` or `vuln.` Additionally, 3 comments are added, to visually separate the three semantic top level elements **Document** Context, and **Vulnerability** (Note: This advisory has no Product Tree instance!):

Example 74:

```
<?xml version="1.0" encoding="UTF-8"?>
<cvrfdoc xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  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:prod="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/prod"
  xmlns:vuln="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/vuln"
  xmlns=="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf"
  >
  <!-- Document wide context information -->
  <DocumentTitle>Apache Struts2 Jakarta Multipart Parser File Upload Code Execution
Vulnerability Affecting Cisco Products<DocumentTitle>
  <DocumentType>Cisco Security Advisory<DocumentType>
  <DocumentPublisher Type="Vendor">
    <ContactDetails>Emergency Support:
+1 877 228 7302 (toll-free within North America)
+1 408 525 6532 (International direct-dial)
Non-emergency Support:
Email: psirt@cisco.com
Support requests that are received via e-mail are typically acknowledged within 48
hours.</ContactDetails>
    <IssuingAuthority>Cisco product security incident response is the responsibility of
the Cisco Product Security Incident Response Team (PSIRT). The Cisco PSIRT is a
dedicated, global team that manages the receipt, investigation, and public reporting of
security vulnerability information that is related to Cisco products and networks. The
on-call Cisco PSIRT works 24x7 with Cisco customers, independent security researchers,
consultants, industry organizations, and other vendors to identify possible security
issues with Cisco products and networks.
More information can be found in Cisco Security Vulnerability Policy available at
http://www.cisco.com/web/about/security/psirt/security_vulnerability_policy.html</cvrf:I
ssuingAuthority>
  </DocumentPublisher>
  <DocumentTracking>
    <Identification>
      <ID>cisco-sa-20170310-struts2</ID>
    </Identification>
    <Status>Interim</Status>
    <Version>1.4</Version>
    <RevisionHistory>
      <Revision>
        <Number>1.0</Number>
        <Date>2017-03-10T20:43:55</Date>
        <Description>Initial public release.</Description>
      </Revision>
      <Revision>
        <Number>1.1</Number>
        <Date>2017-03-11T23:37:26</Date>
        <Description>Updated product lists.</Description>
      </Revision>
      <Revision>
        <Number>1.2</Number>
        <Date>2017-03-13T00:06:20</Date>
        <Description>Updated product lists.</Description>
      </Revision>
      <Revision>
        <Number>1.3</Number>
        <Date>2017-03-13T22:24:49</Date>
        <Description>Updated product lists.</Description>
      </Revision>
      <Revision>
        <Number>1.4</Number>
        <Date>2017-03-14T21:03:12</Date>
        <Description>Updated product lists.</Description>
      </Revision>
    </RevisionHistory>
    <InitialReleaseDate>2017-03-10T19:30:00</InitialReleaseDate>
    <CurrentReleaseDate>2017-03-14T21:03:12</CurrentReleaseDate>
    <Generator>
      <Engine>TVCE</Engine>
    </Generator>
  </DocumentTracking>
  <DocumentNotes>
    <Note Title="Summary" Type="General" Ordinal="1">On March 6, 2017, Apache disclosed
a vulnerability in the Jakarta multipart parser used in Apache Struts2 that could allow
an attacker to execute commands remotely on the targeted system using a crafted Content-
Type header value.

This vulnerability has been assigned CVE-ID CVE-2017-5638.

This advisory is available at the following link:
https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20170310-
struts2 ["https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-
sa-20170310-struts2"]</Note>
    <Note Title="CVSS 3.0 Notice" Type="Other" Ordinal="2">Although CVRF version 1.1
does not support CVSS version 3, the CVSS score in this CVRF file is a CVSSv3 base and
temporal score, as Cisco is now scoring vulnerabilities in CVSSv3.</Note>
  </DocumentNotes>
  <DocumentReferences>
    <Reference Type="Self">
      <URL>https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-
sa-20170310-struts2</URL>
      <Description>Apache Struts2 Jakarta Multipart Parser File Upload Code Execution
Vulnerability Affecting Cisco Products</Description>
    </Reference>
  </DocumentReferences>
  <!-- Vulnerability section -->
  <Vulnerability Ordinal="1"
   xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/vuln">
    <Title>Apache Struts Jakarta Multipart Parser File Upload Code Execution
Vulnerability</Title>
    <Notes>
      <Note Title="Summary" Type="Summary" Ordinal="1">A vulnerability in the Jakarta
multipart parser of Apache Struts could allow an unauthenticated, remote attacker to
execute arbitrary code on an affected system.



The vulnerability is due to improper handling of the Content-Type header value when
performing a file upload based on the Jakarta multipart parser of the affected software.
An attacker could exploit this vulnerability by persuading a targeted user to upload a
malicious file. Once the Jakarta multipart parser of the affected application uploads
the file, the attacker could have the ability to execute arbitrary code.</Note>
    </Notes>
    <CVE>CVE-2017-5638</CVE>
    <Remediations>
      <Remediation Type="Workaround">
        <Description>Any workarounds, when available, are documented in the Cisco bugs,
which are accessible through the Cisco Bug Search Tool
["https://bst.cloudapps.cisco.com/bugsearch/bug/BUGID"].</Description>
      </Remediation>
    </Remediations>
    <References>
      <Reference Type="Self">

<vuln:URL>https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-
sa-20170310-struts2</vuln:URL>
        <Description>Apache Struts2 Jakarta Multipart Parser File Upload Code Execution
Vulnerability Affecting Cisco Products</Description>
      </Reference>
    </References>
  </Vulnerability>
  <!-- No more elements to follow -->
</cvrfdoc>
```

E.4 Sample Security Advisory D

Minimal valid CSAF CVRF version 1.2 document (neither **Product Tree** nor **Vulnerability** noted, but nevertheless well-formed and valid):

```
<?xml version="1.0" encoding="UTF-8"?>
<cvrfdoc xmlns:cvrf="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf">
  <DocumentTitle>DocumentTitle0</DocumentTitle>
  <DocumentType>DocumentType0</DocumentType>
  <DocumentPublisher Type="Vendor">
  </DocumentPublisher>
  <DocumentTracking>
    <Identification>
      <ID>ID0</ID>
    </Identification>
    <Status>Draft</Status>
    <Version>1</Version>
    <RevisionHistory>
      <Revision>
        <Number>1.0</Number>
        <Date>2038-05-04T18:13:51.0</Date>
        <Description>Something wrong with some product</Description>
      </Revision>
      <Revision>
        <Number>1.1</Number>
        <Date>2038-05-04T18:13:52.0</Date>
        <Description>We excluded some products, but still in the fog</Description>
      </Revision>
    </RevisionHistory>
    <InitialReleaseDate>2038-05-04T18:13:51.0</InitialReleaseDate>
    <CurrentReleaseDate>2038-05-04T18:13:52.0</CurrentReleaseDate>
  </DocumentTracking>
</cvrfdoc>
```

E.5 Sample Security Advisory E

A “Minimal Viable Product” like fictitious sample valid CSAF CVRF version 1.2 document is given in the following example.

The vendor is assumed to be named `ACME Inc.`, and DNS-wise only hosted on a subdomain of `example.com` i.e. https://acme.example.com/ and per time zone offset 6 hours “behind” UTC.

A product `foo` is declared to be available on platforms `bar` and `baz` alike and in version `1.9` and `2.1`.

The CVE is made up and uses the new format (as it exceeds the historic 9999 limit of early years, in the hope, that in 2017 there will never be a real `CVE-2017-99999`.

No CWE is given. A CVSS Score is given in CVSSv3 and this is repeated in the human readable portions (maybe because the vendor uses this as accessibility feature).

Some possible combinations of vulnerability and mitigation states are realized i.e.:

- Product `foo` is always known to be affected on platform `bar`.
- Product `foo` is always known not to be affected on platform `baz`.

A remediation page is offered at https://acme.example.com/sa/acme-2017-42-1-1.html an everyone is entitled, but this fact is written in Italian: “`Tutte le persone su questo pianeta`” (English: “All people on this planet”).

The source of the CSAF CVRF version 1.2 security advisory (XML) is offered at https://acme.example.com/sa/acme-2017-42-1-1.xml and an also fictitious variant (for now vendor private) in JSON at https://acme.example.com/sa/acme-2017-42-1-1.json.

Note also the two revisions (in total), maybe the initial release of the security advisory did not take into account, that the product when installed on the `baz` platform is unaffected, or there was given the wrong exploit byte value (not the current `0x42` without access to the initial revision, and no explanation in the release notes, the world will never know (only that something has been corrected).

The acknowledgment was maybe chosen by the company lawyers to not list the real name of the external contributor where the below sample content states “`Some One (not to be named explicitly)`” – and no dummy Organization is added.

The complexity of the Affected/Non-Affected mixture may have forced the publisher to not take the internal usual product IDs but instead append the interesting dimension in a simple serialization as: “`-on-bar`” and “`-on-baz`” respectively. In a real life advisory, these simple platform names might have become fully fledged product entries themselves with relationship link elements between them.

Albeit speculating here, there might have been some time pressure on the publishing vendor `Acme Inc.` of the products `foo` version `1.9` and `2.1`, as sending the byte `0x42` to some port seems to be cheap on the attacker side … at least for the effect of it, when the products are hosted on platform `baz`.

Example 75:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<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"
   xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/cvrf">
   <!-- Document wide context information -->
   <DocumentTitle xml:lang="en">Acme Security Advisory for foo on bar - March 2017 -
CSAF CVRF</DocumentTitle>
   <DocumentType xml:lang="en">Acme Security Advisory</DocumentType>
   <DocumentPublisher Type="Vendor"/>
   <DocumentTracking>
      <Identification>
         <ID>acme-2017-42</ID>
      </Identification>
      <Status>Final</Status>
      <Version>1.0</Version>
      <RevisionHistory>
         <Revision>
            <Number>1.0</Number>
            <Date>2017-03-17T12:34:56-06:00</Date>
            <Description>Initial Distribution</Description>
         </Revision>
         <Revision>
            <Number>1.1</Number>
            <Date>2017-03-18T01:23:45-06:00</Date>
            <Description>Corrected Distribution</Description>
         </Revision>
      </RevisionHistory>
      <InitialReleaseDate>2017-01-17T12:34:56-06:00</InitialReleaseDate>
      <CurrentReleaseDate>2017-01-18T01:23:34-06:00</CurrentReleaseDate>
   </DocumentTracking>
   <DocumentNotes>
      <Note Audience="All" Ordinal="1" Title="Summary" Type="Summary" xml:lang="en">
         This document contains descriptions of Acme product security vulnerabilities
with details on impacted and non-impacted platform product combinations.
         Additional information regarding these vulnerabilities including fix
distribution information can be found at the Acme sites referenced in this
document.</Note>
   </DocumentNotes>
   <DocumentDistribution>This document is published at:
https://acme.example.com/sa/acme-2017-42-1-1.xml</DocumentDistribution>
   <DocumentReferences>
      <Reference Type="External">
         <URL>https://acme.example.com/sa/acme-2017-42-1-1.json</URL>
         <Description>URL to JSON version of Advisory</Description>
      </Reference>
   </DocumentReferences>
   <Acknowledgments>
      <Acknowledgment>
         <Name>Some One (not to be named explicitly)</Name>
      </Acknowledgment>
      <Acknowledgment>
         <Name>Jane Employee</Name>
         <Organization>Acme Inc.</Organization>
      </Acknowledgment>
   </Acknowledgments>
   <!-- Product tree section -->
   <ProductTree xmlns="http://docs.oasis-open.org/csaf/ns/csaf-cvrf/v1.2/prod">
      <Branch Name="Acme" Type="Vendor">
         <Branch Name="Acme Things" Type="Product Family">
            <Branch Name="Things On bar" Type="Product Name">
               <Branch Name="1.9" Type="Product Version">
                  <FullProductName ProductID="AC-FOO-1.9-on-bar">Foo 1.9 on
bar</FullProductName>
               </Branch>
               <Branch Name="2.1" Type="Product Version">
                  <FullProductName ProductID="AC-FOO-2.1-on-bar">Foo 2.1 on
bar</FullProductName>
               </Branch>
            </Branch>
            <Branch Name="Things On baz" Type="Product Name">
               <Branch Name="1.9" Type="Product Version">
                  <FullProductName ProductID="AC-FOO-1.9-on-baz">Foo 1.9 on
baz</FullProductName>
               </Branch>
               <Branch Name="2.1" Type="Product Version">
                  <FullProductName ProductID="AC-FOO-2.1-on-baz">Foo 2.1 on
baz</FullProductName>
               </Branch>
            </Branch>
         </Branch>
       </Branch>
   </ProductTree>
   <!-- Vulnerability sections -->
   <Vulnerability Ordinal="1" xmlns="http://docs.oasis-open.org/csaf/ns/csaf-
cvrf/v1.2/vuln">
      <Title>Vulnerability in the TCP component of Acme foo (CVE-2017-99999)</Title>
      <Notes>
         <Note Audience="All" Ordinal="1" Title="Details" Type="Details">
            Vulnerability in the TCP component of Acme foo.
            Supported versions that are affected are 1.9, and 2.0 when installed on bar
but not affected when on baz.
            Easily exploitable vulnerability allows unauthenticated attacker with
network access via a single 0x42 value payload byte to compromise Acme foo.
            Successful attacks of this vulnerability can result in unauthorized read
access to a subset of Acme foo accessible data and unauthorized ability to cause a
complete denial of service (DOS) of Acme foo.
            CVSS 3.0 Base Score 9.8 (Confidentiality and Availability impacts).
            CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H).</Note>
      </Notes>
      <Involvements>
         <Involvement Party="Vendor" Status="Completed">
            <Description>Fix has been released</Description>
         </Involvement>
      </Involvements>
      <CVE>CVE-2017-99999</CVE>
      <ProductStatuses>
         <Status Type="Known Affected">
            <ProductID>AC-FOO-1.9-on-bar</ProductID>
            <ProductID>AC-FOO-2.1-on-bar</ProductID>
         </Status>
         <Status Type="Known Not Affected">
            <ProductID>AC-FOO-1.9-on-baz</ProductID>
            <ProductID>AC-FOO-2.1-on-baz</ProductID>
         </Status>
      </ProductStatuses>
      <CVSSScoreSets>
         <ScoreSetV3>
            <BaseScoreV3>9.8</BaseScoreV3>
            <VectorV3>CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H</VectorV3>
         </ScoreSetV3>
      </CVSSScoreSets>
      <Remediations>
         <Remediation Type="Vendor Fix">
            <Description>acme-2017-42</Description>
            <Entitlement xml:lang="it">Tutte le persone su questo pianeta</Entitlement>
            <URL>https://acme.example.com/sa/acme-2017-42-1-1.html</URL>
            <ProductID>AC-FOO-1.9-on-bar</ProductID>
            <ProductID>AC-FOO-2.1-on-bar</ProductID>
         </Remediation>
      </Remediations>
   </Vulnerability>
   <!-- No more elements to follow -->
</cvrfdoc>
```

Appendix F. Index

Appendix G. Revision History

Revision	Date	Editor	Changes Made
Working Draft 01	2017-03-24	Stefan Hagen	Combined and migrated CVRF version 1.1 contribution to OASIS CSAF CVRF version 1.2 and added feedback from TC members.
Committee Specification Draft 01	2017-05-24	Stefan Hagen	Package for public review.