The Missing Manual: CVRF 1.1

Mike Schiffman <mschiffm@cisco.com>
The Internet Consortium for Advancement of Security on the Internet

April 2012

Prolegomenon

In this whitepaper you will learn about some of the design decisions behind the 1.1 release of the Common Vulnerability Reporting Framework. Particular attention is paid to explaining some of the required elements and the Product Tree. After those tasty tidbits, we will convert a recent Cisco security advisory into well-formed and valid CVRF document. To close, you are treated to some of the items on the docket for future versions of CVRF. It bears mentioning that this paper is not meant to be an exhaustive explanation of the CVRF schemata. It is a rather capricious, if somewhat disorganized look at some outliers that aren't fully explained elsewhere. It is assumed the reader has a working knowledge of the Common Vulnerability Reporting Framework and of XML.

CVRF: Helping industry get their ducks in a row since 2011

After the release of CVRF 1.0 in May of 2011, I found myself at a dinner with some technical folks. Security automation came up, followed by a brief discussion of CVRF. I somehow managed to coin a phrase that quite succinctly captured what CVRF did, does, and will do. So happy was I that I scribbled it down on a napkin: "*...the goal of CVRF is to homogenize and automate the creation and consumption of vulnerability documentation.*" Wow, that's just a profoundly simple yet perfectly encompassing statement. Simple, profound and complete enough to be a mission statement! It's so important; I think we should restate it and perhaps extol some of the keywords as shown in Figure 1:

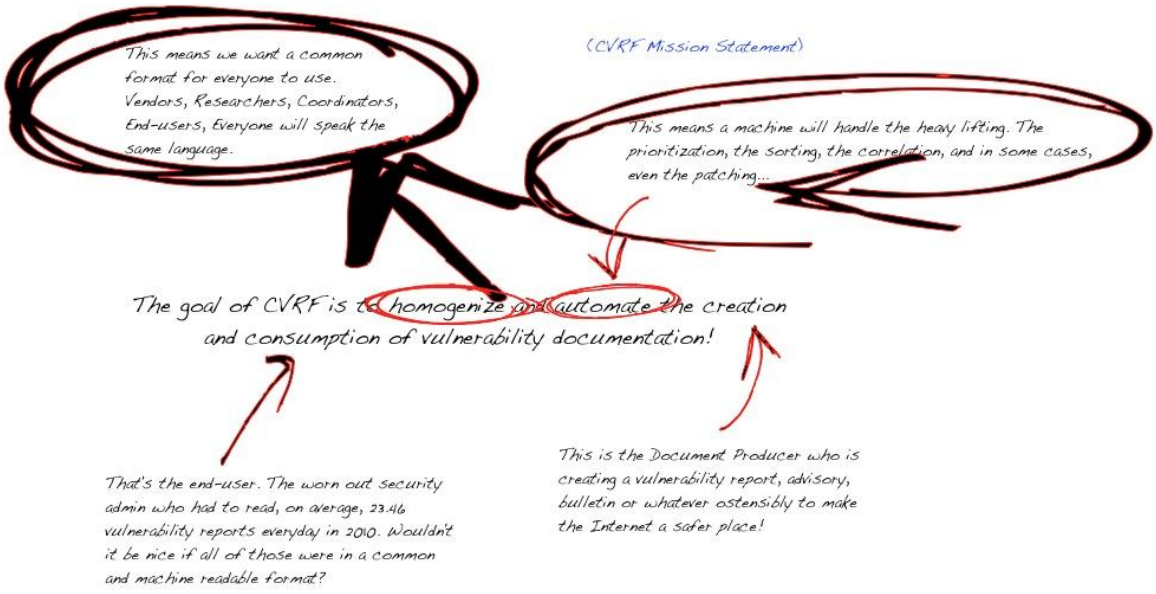

FIGURE 1.

Like so much in life, it is simple in concept yet complex in execution. It has taken a group of intrepid ICASI-based adventurers over three years to stich together your CVRF. To be sure, it has evolved a great deal since we first embarked, but always with the same goal in mind. Currently CVRF is positioned to be the de facto standard in vulnerability documentation automation. CVRF 1.0 was the initial foray into this space. Now, almost one year later, the follow-up, CVRF 1.1 is a major leap forward and represents the current best-of-breed solution. So let's get on with the show and explore a few core CVRF concepts.

Mind Mapping

A primary tool of development for CVRF is the mind map. Mind maps are wonderful tools for representing ideas and brainstorming in real-time. As we turn the corner on three years of CVRF

development, the principal way CVRF is shown to the World is through a mind map diagram. The legend is shown in Table 1.

Node Type	**Meaning**
Bubble	Element
Underline	Attribute
Font Color	
Red	Required
Black	Normal
Blue	Choice
Font Type	
Normal	Normal
Bold	Container

TABLE 1.

CVRF is shown in full in Figure 2.

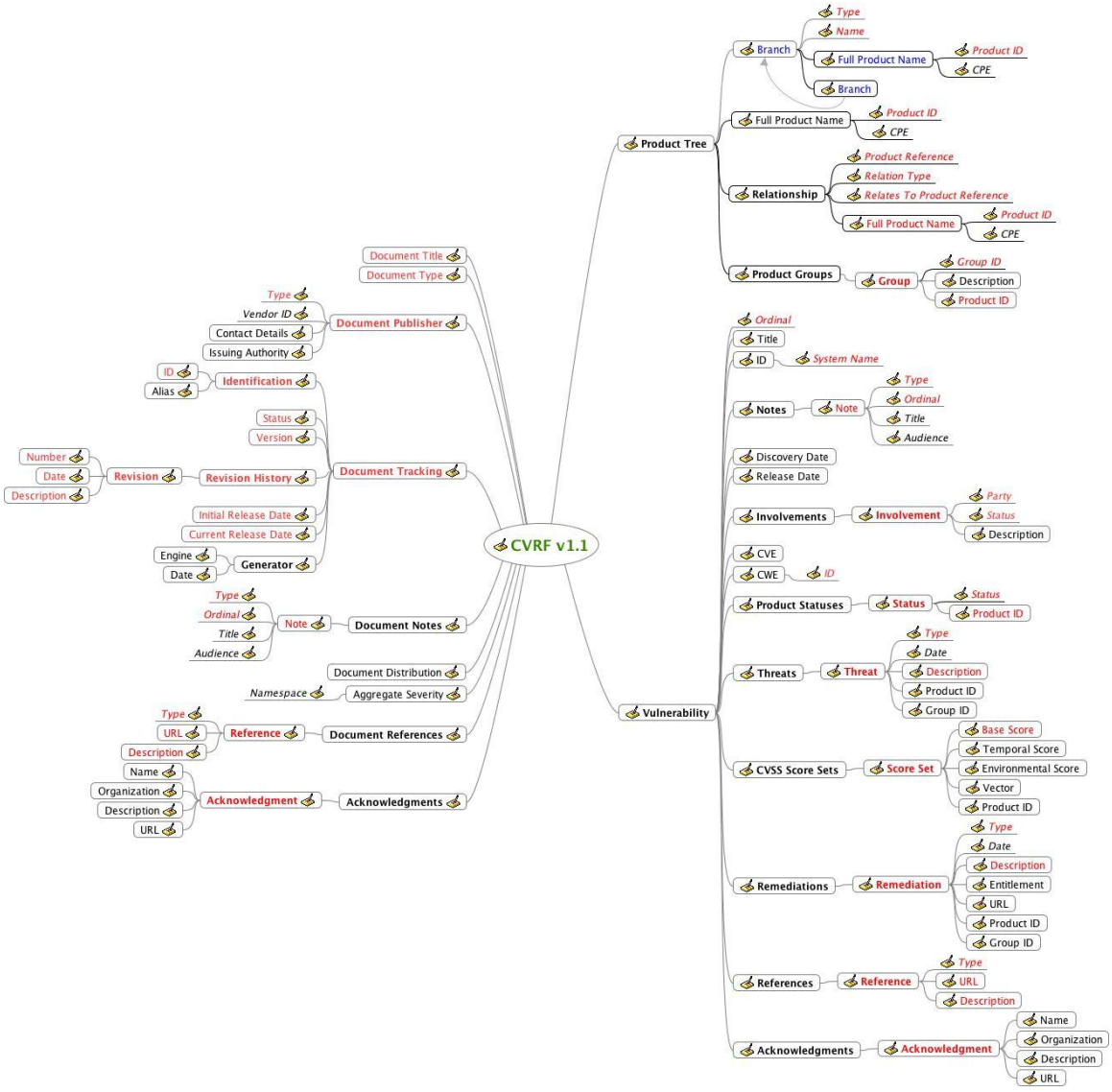

FIGURE 2.

Let's Be Objective

One of the major design tenets of CVRF was objectivity over subjectivity. We understand that computer security can be an emotional topic at times, but the framework used to convey information pertaining to that subject should be free from opinion and bias; it should just report facts. With this in mind, CVRF was grown in a lab. Each element was named and defined as objectively as possible. We wanted people, regardless of their background, to be able to understand CVRF the same way.

Data-Driven vs. Presentation-Driven

Another design decision made mid-way through the engineering process was for CVRF to adhere to a data-driven model. This is to say that CVRF is only concerned with packaging up and structuring the data, without regard for how that data will be presented to the end-user. There are no presentation markup elements (for example, HTML's `<b>…</b>` or `<i>…</i>`) included in CVRF.

CVRF Doesn't Do Anything

CVRF is a language. It is a structured way for one machine to communicate with another machine about a topic their human operators find very interesting: information system vulnerabilities. CVRF was built using XML: XML is a meta-language – it doesn't actually do anything. CVRF doesn't actually do anything, either. It is a static framework used to structure, store, and transport information. CVRF is merely a structured container for information about information system vulnerabilities. How that information is ultimately transported, parsed, or used is completely up to its users.

What's Required?

Another design decision of CVRF revolved around the consensus that the document must contain a modicum of mandatory, root-level elements. This is done to ensure a solid and consistent foundation upon which to construct documents. The required elements are shown in Figure 3.

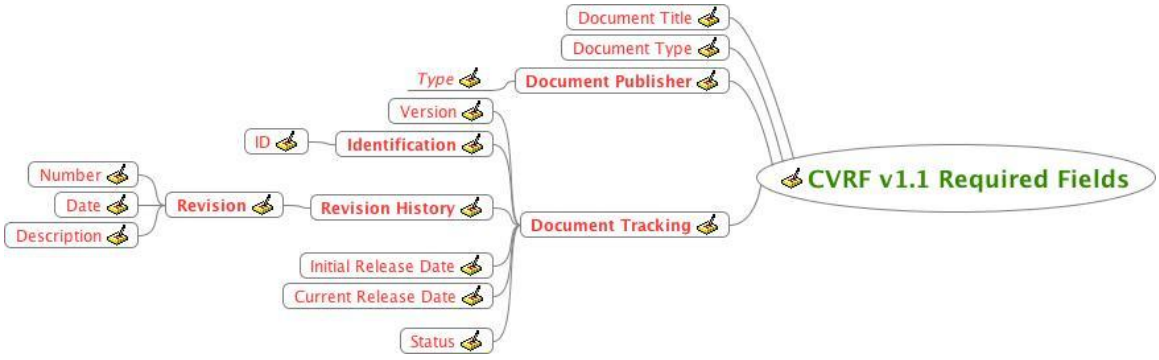

FIGURE 3.

Container Notation

Throughout CVRF, you will find optional container elements that use a plural term as the identifier (e.g. "Threats"), followed by a child container that uses the singular version of the same identifier (e.g. "Threat"). The parent container is usually optional (MinOccurs=0) and cannot occur more than once (MaxOccurs=1), while the child container is required (MinOccurs=1) and can have unlimited instances (MaxOccurs=unbounded).

For example: the Threats container can contain multiple Threat containers. If no threats are to be documented, the Threats container should be omitted altogether. However, if a Threats container is defined, you need to specify at least one (more are allowed) child Threat container as per the following:

```
<Threats>
    <Threat Type="Impact">
        <Description>
        Successful exploit this vulnerability results in every byte
        on the target network being replaced by the ones' compliment of
        itself.
        </Description>
    </Threat>
    <Threat Type="Exploit Status">
        <Description>
        The exploit is being carried as part of the OKBDGLOBAL.A1 worm
        and is in widespread use.
        </Description>
    </Threat>
    <Threat Type="Target Set">
        <Description>
        Every network known to man.
        </Description>
    </Threat>
</Threats>
```

Character Data

CVRF has several unrestricted "string" typed fields that assumed to be human consumable. These fields allow you (as a document producer) to input arbitrary character data without constraint around length or content. The best common practice here is quite simply to ensure this character data is free from markup language tags (XML, HTML, XHTML, etc.). However, in some cases, you will find yourself faced with a situation where you are forced to copy data into a CVRF document that contains markup. Often in a Notes or Entitlement element, it is sometimes unavoidable. The general guidance here is the following:

1. Avoid including markup data in string fields. If it is unavoidable see #2.
2. Escape the special markup characters (for example & becomes &).
3. If large blocks of markup are to be used, employ CDATA tags.

The XML construct CDATA instructs the parser to ignore whatever text is enclosed inside of it. To be clear, when you want to migrate an existing document as-is without needing to worry about encoding XML entities, either escape each special character or use CDATA. Please note that if you want the parser to be able to interpret the text within string sections CDATA can make it difficult for document consumers to parse and transform CVRF documents.

An exception to this rule is when you want to include un-rendered markup elements inside a string element for illustrative purposes. An example:

```
<Notes>
    <Note Type="Details" Ordinal="120" Title="Quick Fix"
          Audience="Technical">
    If you change the Tomcat web.xml to the following, the problem will
    go away:
    <![CDATA[
    <parameter timeout="0" />
    ]]>
    </Note>
</Notes>
```

A CDATA example is below in the Praxis section.

Product References

Many elements within a CVRF document allow (or require) you to specify one or more Product ID or Group ID references, indicating that the current element applies to one or more products. In most cases, specifying a Product ID or Group ID is optional, and as such omitting the ID indicates that the current entry is general or non-specific. If an element applies to more than one product, you can choose whether to list all corresponding Product ID values separately, or whether to define a group first and then reference the corresponding Group ID – it's up to you.

In the Vulnerability section, the following containers allow product referencing: Products/Product, Threats/Threat, CVSS Score Sets/Score Set, and Remediations/Remediation. Additionally, all of these containers follow the previously mentioned conventions for plural/singular container notation.

The Product Tree

The Product Tree is a new construct for CVRF 1.1 and is a total redesign in how CVRF enumerates products in a vulnerability. In earlier versions, after multiple rounds of peer review, we noticed that people were having several issues with the way products were specified:

- **It was inefficient:** It violated "Don't Repeat Yourself" model of software development. Since each product was encapsulated inside the Vulnerability container, if you had multiple vulnerabilities in the same product, you would have to repeat the same XML code in each Vulnerability container.
- **It was confusing:** You could have an Affected Platform and/or an Affected Release that were actually fixed or recommended versions.
- **It didn't enforce machine-parseable names:** Affected Platform and Affected Release each had Name attributes that were strings with recommendations, but not restrictions, as to how to best populate them.

To solve these problems, we scrapped Product Family and the Affected elements and created a hierarchical Product Tree as shown in Figure 4.

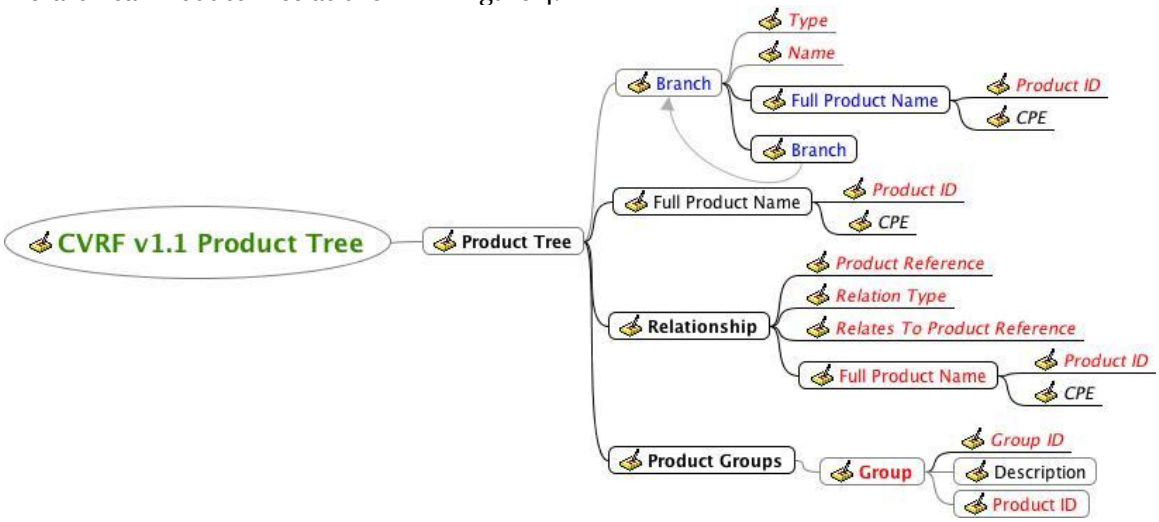

FIGURE 4.

On a high-level, it contains all of the fully qualified product names that can be referenced elsewhere in a CVRF document. It solved all of the problems above.

- **It is efficient:** Because the Product Tree is an external to the Vulnerability container(s), it can be specified once within the document and referred to many times. Each Vulnerability container can now contain a reference to the same product rather than having to redundantly list each product over again.
- **Product status is clear:** Products aren't assigned a status inside the Product Tree. A product's susceptibility or inoculation status will be different across vulnerabilities. It is now left up to Vulnerability/Products/Product to assign a status ("Known Affected", "First Fixed", and so on).
- **Machine consumable naming is permitted:** Using the branched method a document producer can create a product name using machine-readable tags combined with human-readable names (see below).

Additionally, it affords the user a way to specify complex relationships between products or components (as a precondition for being vulnerable) using the Relationship container.

Praxis: Converting an existing document to CVRF

Now it’s time for some XML! Let's take what you've learned and manually convert the Cisco RVS4000 and WRVS4400N Web Management Interface Vulnerabilities security advisory (http://www.cisco.com/en/US/products/csa/cisco-sa-20110525-rvs4000.html) into a CVRF document. Please note that this process is meant to be instructive and somewhat of a stream-of-consciousness-narrative of how to manually build your first CVRF document. It is expected that,

by and large, this process would itself be automated and CVRF document producers would have in-house code to parse their own documents and emit CVRF.

From here on out, the original Cisco security advisory HTML document will be referred to as "SA" while the CVRF document will be referred to as "CVRF document" or some derivative. Remember that an XML schema defines the structure and content of the elements inside an XML document, as well as the order they can be in. This means that we might have to hunt around the SA to find all of the elements we want or need, but we must conform to the order set out by the schema. Before starting this section, have a quick look at the SA and be comfortable with its format and sections.

Disclaimer: What follows should be considered a tutorial only. Not all of the SA is translated into the CVRF document and is by no means a Cisco PSIRT authored or endorsed document.

Root (Document) Level

1) We'll start with a standard XML prologue specifying the version, encoding and the namespace for the schema:

```
<?xml version="1.0" encoding="UTF-8"?> <cvrfdoc
xmlns="http://www.icasi.org/CVRF/schema/cvrf/1.1"
xmlns:cvrf="http://www.icasi.org/CVRF/schema/cvrf/1.1">
```

2) We'll choose a title for our CVRF document. This is a required, human-readable field, and we'll simply copy over the SA title:

```
    <DocumentTitle>
    Cisco Security Advisory: Cisco RVS4000 and WRVS4400N Web Management
    Interface Vulnerabilities
    </DocumentTitle>
```

3) Next we add the required field, Document Type, another human-readable field that we'll call "Security Advisory":

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

4) Next we add the required field, Document Publisher. All that is required is the type, and we'll fill that in as Vendor. Note that unlike the previous fields, we can only choose from five different values (vendor, discoverer, coordinator, user, other):

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

5) The large, monolithic and required Document Tracking container is next. Comments are inline.

```
    <DocumentTracking>
```


5a) The required Identification container holds one ID string and additional optional Alias strings that identify the CVRF document. The required ID element, is set to the Cisco specific identifier for the SA which is derived from the "Advisory ID" field of "cisco-sa-20110525-rvs4000".

```
        <Identification>
            <ID>cisco-sa-20110525-rvs4000</ID>
        </Identification>
```

5b) The required Status element is set from the "Status of this Notice" field in the SA which is "Final".

```
       <Status>Final</Status>
```

5c) The required Version element is set from the "Revision" field of the SA which is "1.1". The Revision History is taken from the SA fields of the same name. All elements are required.

```
        <Version>1.1</Version>
        <RevisionHistory>
            <Revision>
                <Number>1.1</Number>
                <Date>2011-06-17T00:00:00+00:00</Date>
                <Description>
                Modified software table to indicate that First Fixed release
                for RVS4000v1 is 1.3.3.5.
                </Description>
            </Revision>
            <Revision>
                <Number>1.0</Number>
                <Date>2011-05-25T00:00:00+00:00</Date>
                <Description>Initial public release.</Description>
            </Revision>
        </RevisionHistory>
```

5d) The required Initial and Current Release Date elements are adapted from the SA fields "Initial Public Release" and "Last Updated".

```
        <InitialReleaseDate>2011-05-25T00:00:00+00:00</InitialReleaseDate>
        <CurrentReleaseDate>2011-06-17T00:00:00+00:00</CurrentReleaseDate>
    </DocumentTracking>
```

6) The next container is Document Notes. This is where we store all freeform human readable text blurbs from around the SA and its use is completely optional. The first Note we create is to hold the SA Summary. The only required attributes are Type, which is "Summary" for the first Note and "Legal Disclaimer" for the second, and Ordinal, which is a locally significant counter used to track all of the Document Notes and we set to "001" and "002". We'll entitle the first Note as "Summary" and the second as "Legal Disclaimer" and include an Audience set to "All" for both.

Both attributes are human readable fields. Finally, we include the XML:lang attribute to indicate the language of both Notes, in this case, English. The use of the XML:lang attribute is completely optional when you're writing in English since that is the default. If you change to a different language, you would then use it to specify which one.

```
<DocumentNotes>
    <Note Title="Summary" Audience="All" Type="Summary" Ordinal="001"
          xml:lang="en">
    Cisco RVS4000 4-port Gigabit Security Routers and Cisco WRVS4400N
    Wireless-N Gigabit Security Routers have several web interface
    vulnerabilities that can be exploited by a remote, unauthenticated
    user. Cisco has released free software updates that address these
    vulnerabilities. Workarounds that mitigate these vulnerabilities are
    available. This advisory is posted at http://tools.cisco.com/security/
    center/content/CiscoSecurityAdvisory/cisco-sa-20110525-rvs4000.
    </Note>
    <Note Title="Legal Disclaimer" Audience="All" Type="Legal Disclaimer"
          Ordinal="002" xml:lang="en">
    THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS AND DOES NOT IMPLY ANY
    KIND OF GUARANTEE OR WARRANTY, INCLUDING THE WARRANTIES OF
    MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE. YOUR USE OF THE
    INFORMATION ON THE DOCUMENT OR MATERIALS LINKED FROM THE DOCUMENT IS AT
    YOUR OWN RISK. CISCO RESERVES THE RIGHT TO CHANGE OR UPDATE THIS
    DOCUMENT AT ANY TIME. CISCO EXPECTS TO UPDATE THIS DOCUMENT AS NEW
    INFORMATION BECOMES AVAILABLE. A stand-alone copy or Paraphrase of the
    text of this document that omits the distribution URL in the following
    section is an uncontrolled copy, and may lack important information or
    contain factual errors.
    </Note>
</DocumentNotes>
```

7) Acknowledgements is an optional container, but since the SA credits the original discoverer, we will include one here.

```
<Acknowledgments>
    <Acknowledgment>
        <Name>Michal Sajdak</Name>
        <Organization>Securitum, Poland</Organization>
    </Acknowledgment>
</Acknowledgments>
```

8) Also optional is the Document References container. For posterity, we will include a single Related Document that refers to the original HTML SA. The Type of reference is self because we're referring to a different representation of the same document. The URL points to the HTML SA (a CRLF is inserted to preserve readability in the whitepaper, but should not be used in an actual CVRF document) and the human readable Description is nominal.


```
    <DocumentReferences>
        <Reference Type="Self">
        <URL>
        http://tools.cisco.com/security/center/
        content/CiscoSecurityAdvisory/cisco-sa-20110525-rvs4000
        </URL>
        <Description xml:lang="en">URL to the html advisory.</Description>
    </Reference>
</DocumentReferences>
```

9) The Document Distribution comes next. We find it in the SA under "Distribution". This is another human readable field and I prefer to remove all non-character data (bullets and such):

```
    <DocumentDistribution>
    This advisory is posted on Cisco's worldwide website at
    http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/
    cisco-sa-20110525-rvs4000 In addition to worldwide web posting, a text
    version of this notice is clear-signed with the Cisco PSIRT PGP key and is
    posted to the following e-mail and Usenet news recipients.
    - cust-security
    - announce@cisco.com
    - first-bulletins@lists.first.org
    - bugtraq@securityfocus.com
    - vulnwatch@vulnwatch.org
    - cisco@spot.colorado.edu
    - cisco-nsp@puck.nether.net
    - full-disclosure@lists.grok.org.uk
    - comp.dcom.sys.cisco@newsgate.cisco.com
    Future updates of this advisory, if any, will be placed on Cisco's
    worldwide website, but may or may not be actively announced on mailing
    lists or newsgroups. Users concerned about this problem are encouraged to
    check the above URL for any updates.
    </DocumentDistribution>
```

Product Tree

Next we will build our Product Tree to contain all of the products we will reference in the Vulnerability section of the CVRF document. After reading the SA, we find in the "Affected Products" section that there are five products affected by the vulnerability and in the "Software Versions and Fixes" section there are three "first fixed" products (two of the affected versions have been end of life'd and are no longer supported):

Affected Products:

1. Cisco RVS4000 Gigabit Security Router version 1
2. Cisco RVS4000 Gigabit Security Router version 2
3. Cisco WRVS4400N Wireless-N Gigabit Security Router version 1.0

4. Cisco WRVS4400N Wireless-N Gigabit Security Router version 1.1
5. Cisco WRVS4400N Wireless-N Gigabit Security Router version 2.0

Fixed Products:

6. Cisco RVS4000 Gigabit Security Router version 1.3.3.5
7. Cisco RVS4000 Gigabit Security Router version 2.0.2.7
8. Cisco WRVS4400N Wireless-N Gigabit Security Router 2.0.2.1

In the following example, we will use the "Branched" method of Product Tree creation to insert the above products. It affords the most machine-readable method of product enumeration (the "Concatenated" method does a similar job, but this SA contains no product dependencies).

1) First element is the Product Tree container that specifies the schema.

```
    <ProductTree xmlns="http://www.icasi.org/CVRF/schema/prod/1.1">
```

2) We then construct each product using the Branched method. Since all of the products are vended by Cisco, our top level Branch container is Type "Vendor" and Name "Cisco". Both attributes are required.

```
        <Branch Type="Vendor" Name="Cisco">
```

3) This level of nesting actually splits. We first handle the "RVS4000" products and we finish up with the "WRVS4400N" products. This is the Product Family and we assign Type and Name accordingly.

```
            <Branch Type="Product Family" Name="RVS4000">
```

4) Next is the canonical name of the product. We assign this Type as Product Name with a Name value of "Gigabit Security Router".

```
                <Branch Type="Product Name" Name="Gigabit Security Router">
```

5) The last branch contains the product version. We assign this Type as Product Version with a Name value of "1". We then terminate this branch with a Full Product Name and our first Product ID. The FullProductName contains the canonical or “friendly” name of the referenced product. In this (and probably most) cases, it is the additive result of the individual branches, or “Cisco RVS4000 Gigabit Security Router version 1”. The ProductID is chosen to be illustrative and demonstrative at “CVRF1.1-PID-0001”. In practice, it can be anything unique to Product Tree.

```
                    <Branch Type="Product Version" Name="1">
                        <FullProductName ProductID="CVRF1.1-PID-0001">
                        Cisco RVS4000 Gigabit Security Router version 1
                        </FullProductName>
                    </Branch>
```


6) The remainder of the Product Tree elements are constructed in the same manner.

```
                <Branch Type="Product Version" Name="2">
                    <FullProductName ProductID="CVRF1.1-PID-0002">
                    Cisco RVS4000 Gigabit Security Router version 2
                    </FullProductName>
                </Branch>
                <Branch Type="Product Version" Name="1.3.3.5">
                    <FullProductName ProductID="CVRF1.1-PID-0006">
                    Cisco RVS4000 Gigabit Security Router version 1.3.3.5
                    </FullProductName>
                </Branch>
                <Branch Type="Product Version" Name="2.0.2.7">
                    <FullProductName ProductID="CVRF1.1-PID-0007">
                    Cisco RVS4000 Gigabit Security Router version 2.0.2.7
                    </FullProductName>
                </Branch>
            </Branch>
        </Branch>
        <Branch Type="Product Family" Name="WRVS4400N">
            <Branch Type="Product Name"
                    Name="Wireless-N Gigabit Security Router">
                <Branch Type="Product Version" Name="1.0”>
                    <FullProductName ProductID="CVRF1.1-PID-0003">
                    Cisco WRVS4400N Wireless-N Gigabit Security Router
                    version 1.0
                    </FullProductName>
                </Branch>
                <Branch Type="Product Version" Name="1.1">
                    <FullProductName ProductID="CVRF1.1-PID-0004">
                    Cisco WRVS4400N Wireless-N Gigabit Security Router
                    version 1.1
                    </FullProductName>
                </Branch>
                <Branch Type="Product Version" Name="2.0">
                    <FullProductName ProductID="CVRF1.1-PID-0005">
                    Cisco WRVS4400N Wireless-N Gigabit Security Router
                    version 2.0
                    </FullProductName>
                </Branch>
                <Branch Type="Product Version" Name="2.0.2.1">
                    <FullProductName ProductID="CVRF1.1-PID-0008">
                    Cisco WRVS4400N Wireless-N Gigabit Security Router
                    version 2.0.2.1
                    </FullProductName>
                </Branch>
            </Branch>
```



```
            </Branch>
        </Branch>
    </ProductTree>
```

Vulnerability

The final section contains the actual vulnerabilities documented in the SA. If we read the "Details" section of the SA, we find that there are three vulnerabilities:

Vulnerabilities:

1. CVE-2011-1645: Retrieval of the configuration file
2. CVE-2011-1646: Root operating system arbitrary command injection by an authenticated attacker
3. CVE-2011-1647: Retrieval of admin SSL certificate private key

We will document each in its own Vulnerability container.

1) First element is the Vulnerability container that has a required attribute, Ordinal.

```
    <Vulnerability Ordinal="1"
                   xmlns="http://www.icasi.org/CVRF/schema/vuln/1.1">
```

2) Next we have the human readable Title. We simply copy the canonical name from the SA.

```
        <Title>Retrieval of the configuration file</Title>
```

3) Next we find the Notes container. Structurally, it is the same as the Document Notes container above. In this case, it contains details about the vulnerability and is typed and titled as such. We see the first usage of CDATA here, as there is an ampersand in the original text (which, from an XML point of view, it tells the parser an escape sequence is forthcoming).

```
        <Notes>
            <Note Type="Details" Ordinal="003" Title="Details" Audience="All">
            <![CDATA[
            The Cisco RVS4000 and WRVS4400N Gigabit Security Routers deliver
            high-speed network access and IPsec VPN capabilities for small
            businesses. They also provides firewall and intrusion prevention
            capabilities. The Cisco RVS4000 and WRVS4400N Gigabit Security
            Routers contains a web management interface vulnerability:
            Retrieval of the configuration file If an administrator of the
            device has previously created a backup  of the configuration, using
            Administration --> Backup & Restore --> Backup, it is possible for
            a remote unauthenticated user to access the backup configuration
            file.  This file contains all configuration
            parameters of the device, including the HTTP  authentication
```



```
            password and VPN pre-shared-keys (PSKs).
            ]]>
            </Note>
        </Notes>
```

4) The release date of the vulnerability is set as the initial release date of the advisory.

```
        <ReleaseDate>2011-05-25T00:00:00+00:00</ReleaseDate>
```

5) The Involvements container houses information about the document producer's engagement status of vulnerability. In this case, the Party is Cisco (the vendor of the product) and their involvement Status is considered Completed (the vulnerability is fully identified and scoped and the remediation process is finished).

```
        <Involvements>
            <Involvement Party="Vendor" Status="Completed">
                <Description>
                Cisco has completed involvement in this vulnerability.
                </Description>
            </Involvement>
        </Involvements>
```

6) We find a CVE number for the vulnerability in the SA's "Details" section.

```
        <CVE>CVE-2011-1645</CVE>
```

7) Now we move on to the Product Statuses. This is where we enumerate the list of products that are directly concerned with vulnerability. The guidance here is to list the affected products followed by the fixed or recommended products.

```
        <Products Statuses>
```

7a) If we look at the "Affected Products/Vulnerable Products" section we will note which products and versions are vulnerable. If we then refer back to the Product Tree, we can enumerate their Product IDs.

```
            <Status Type="Known Affected">
                <ProductID>CVRF1.1-PID-0001</ProductID>
                <ProductID>CVRF1.1-PID-0002</ProductID>
                <ProductID>CVRF1.1-PID-0003</ProductID>
                <ProductID>CVRF1.1-PID-0004</ProductID>
                <ProductID>CVRF1.1-PID-0005</ProductID>
            </Status>
```

7b) If we look at the "Software Versions and Fixes" section we will note which products and versions are fixed. If we then refer back to the Product Tree, we can enumerate their Product IDs.


```
            <Status Type="First Fixed">
                <ProductID>CVRF1.1-PID-0006</ProductID>
                <ProductID>CVRF1.1-PID-0007</ProductID>
                <ProductID>CVRF1.1-PID-0008</ProductID>
            </Status>
        </Product Statuses>
```

8) Next we happen upon the Threats container. If we peruse the "Impact" section of the SA we find a description. That's fodder for our Impact typed Threat container. The omission of any Product or Group IDs means this threat applies to all of the vulnerable products.

```
        <Threats>
            <Threat Type="Impact">
                <Description>
                Successful exploitation of the vulnerability may result in
                execution of arbitrary commands on the device by an
                authenticated user or retrieval of configuration files and
                private keys by an unauthenticated user. The configuration
                files contain sensitive information in text, such as the HTTP
                passwords and PSKs. The retrieval of the certificates may aid
                in further attacks.
                </Description>
            </Threat>
        </Threats>
```

9) The next container we come across is the CVSS Score Sets. This stuff can be found in the "Vulnerability Scoring Details" section. A Score Set will always have a Base Score, but the Temporal and Environmental scores as well as the Vector, are optional. In this case, we have a temporal score and a Vector and we fill them in accordingly.

```
        <CVSSScoreSets>
            <ScoreSet>
                <BaseScore>9.3</BaseScore>
                <TemporalScore>7.7</TemporalScore>
                <Vector>
                AV:N/AC:M/Au:N/C:C/I:C/A:C/E:F/RL:OF/RC:C
                /CDP:ND/TD:ND/CR:ND/IR:ND/AR:ND
                </Vector>
            </ScoreSet>
        </CVSSScoreSets>
```

10) The final section for Vulnerability is the Remediation container.

```
        <Remediations>
```

10a) The first step here is the determine the type of Remediation. We learn from the "Software Versions and Fixes" section that there is an official vendor fix, and we assign that as the Type.


```
        <Remediation Type="Vendor Fix">
```

10b) The mandatory Description element is created from the same section.

```
            <Description>
            The official fix from Cisco is to upgrade to the latest
            software release, the first fixed version is 1.3.3.5.
            In all cases, customers should exercise caution to be certain
            the devices to be upgraded contain sufficient memory and that
            current hardware and software configurations will continue to
            be supported properly by the new release. If the information is
            not clear, contact the Cisco Small Business Support Center or
            your contracted maintenance provider for assistance.
            </Description>
```

10c) Entitlement is optional to the Remediation container, but likely a legal mandate by the Cisco powers that be. It is copied from the "Obtaining Fixed Software" section.

```
            <Entitlement>
            Cisco has released free software updates that address this
            vulnerability. Prior to deploying software, customers should
            consult their maintenance provider or check the software for
            feature set compatibility and known issues specific to their
            environment. Customers may only install and expect support for
            the feature sets they have purchased. By installing,
            downloading, accessing or otherwise using such software
            upgrades, customers agree to be bound by the terms of Cisco's
            software license terms found at http://www.cisco.com/en/US/
            docs/general/warranty/English/EU1KEN_.html , or as otherwise
            set forth at Cisco.com Downloads at http://www.cisco.com/
            public/sw-center/sw-usingswc.shtml . Do not contact
            psirt@cisco.com or security-alert@cisco.com for software
            upgrades. Customers should obtain upgraded software through
            their regular update channels. For most customers, this means
            that upgrades should be obtained through the Software Center on
            Cisco's worldwide website at http://www.cisco.com If the
            information is not clear, please contact the Cisco Small
            Business Support Center or your contracted maintenance provider
            for assistance. Small Business Support Center contacts are as
            follows. +1 866 606 1866 (toll free from within North America)
            +1 408 418 1866 (toll call from anywhere in the world)
            Customers should have their product serial number available.
            Refer to http://www.cisco.com/en/US/support/
            tsd_cisco_small_business_support_center_contacts.html for
            additional support contact information, including localized
            telephone numbers, and instructions and e-mail addresses for
```



```
            use in various languages.
            </Entitlement>
```

10d) URL to the fixed software is optional but included here as it is specified by the SA in the "Software Versions and Fixes" section. Note the use of the ampersand escape sequence to specify the "&" in the query string portion of the URL.

```
            <URL>
            http://www.cisco.com/cisco/software/
            type.html?mdfid=282414013&amp;flowid=787
            </URL>
```

10e) Finally we come to the last element in the Remediation section, the optional Product ID. Inclusion of this (these) element(s) constrains the Remediation container to apply to only the product(s) listed. Exclusion of the element implies that the Remediation is general or applies to all vulnerable products.

```
            <ProductID>CVRF1.1-PID-0001</ProductID>
        </Remediation>
```

10f) Now we can move on to the next two Remediation containers, which are constructed similar to the first, the only notable differences being the Description, the Product ID and the URL of the last one. All of this information is in the same section "Obtaining Fixed Software". The entitlement is omitted to keep the code short, sweet and DRY.

```
        <Remediation Type="Vendor Fix">
            <Description>
            The official fix from Cisco is to upgrade to the latest
            software release, the first fixed version is 2.0.2.7.
            </Description>
            <Entitlement>
            ...
            </Entitlement>
            <URL>
            http://www.cisco.com/cisco/software/
            type.html?mdfid=282414013&amp;flowid=787
            </URL>
            <ProductID>CVRF1.1-PID-0002</ProductID>
        </Remediation>
        <Remediation Type="Vendor Fix">
            <Description>
            The official fix from Cisco is to upgrade to the latest
            software release, the first fixed version is 2.0.2.1.
            </Description>
            <Entitlement>
            ...
```



```
                </Entitlement>

                <URL>
                http://www.cisco.com/cisco/software/type.html?mdfid=282414016
                </URL>
```

10g) Note we have multiple Product ID's here.

```
                <ProductID>CVRF1.1-PID-0003</ProductID>
                <ProductID>CVRF1.1-PID-0004</ProductID>
                <ProductID>CVRF1.1-PID-0005</ProductID>
            </Remediation>
```

10h) This Remediation container is slightly different because it holds the workaround discussed in the SA in the section entitled "Workarounds" and it applies to all of the vulnerable products.

```
            <Remediation Type="Workaround">
                <Description>
                Disable remote management: Caution: Do not disable remote
                management if you manage the device via the WAN connection.
                Doing so will result in loss of management connectivity to the
                device. Remote Management is disabled by default. If it is
                enabled, administrators can disable it using the Firewall >
                Basic Settings screen. Change the setting for the field
                "Remote Management" to "Disabled". Disabling remote management
                limits the exposure of the vulnerabilities to those on the
                local LAN. Limit remote management access to specific IP
                addresses: If remote management is required, harden the device
                so that it can be accessed only by certain IP addresses,
                rather than the default setting of "any". By entering the
                configuration screen at Firewall --> Basic Settings, an
                administrator can change the Remote IP address field to ensure
                only devices with the specified IP addresses can access the
                device. Remove all backup configuration files from the device:
                Rebooting the device after performing a configuration backup,
                will remove the configuration file from the system so that it
                can not be retrieved by an unauthenticated user.
                </Description>
                <ProductID>CVRF1.1-PID-0001</ProductID>
                <ProductID>CVRF1.1-PID-0002</ProductID>
                <ProductID>CVRF1.1-PID-0003</ProductID>
                <ProductID>CVRF1.1-PID-0004</ProductID>
                <ProductID>CVRF1.1-PID-0005</ProductID>
            </Remediation>
        </Remediations>
    </Vulnerability>
```


11) The remaining two vulnerabilities are quite similar to the first in terms of construction and content. To close out our CVRF document, for brevity's sake, they are left to you as an exercise.

```
    <Vulnerability Ordinal="2"
                   xmlns="http://www.icasi.org/CVRF/schema/vuln/1.1">
        <Title>Root operating system arbitrary command injection by an
               authenticated attacker
        </Title>
    ...
    </Vulnerability>
    <Vulnerability Ordinal="3"
                   xmlns="http://www.icasi.org/CVRF/schema/vuln/1.1">
        <Title>Retrieval of admin SSL certificate private key</Title>
    ...
    </Vulnerability>
</cvrfdoc>
```

Looking Ahead: What's Next for CVRF?

As CVRF evolves, it will change and grow as per the feedback from its users. The follow items are on the docket for inclusion in subsequent release(s) of CVRF. The list is not exhaustive and subject to change.

- **XML Security**: Support for the signing and encrypting of XML documents as per the W3C recommendations on XML Signature Syntax and Processing.
- **Producer Specific Tags**: Support for producer specific elements that would enable a document producer to include their own XML tags to add functionality to CVRF. The working group is carefully discussing this feature. We want to avoid the situation where it would enable the creation of new CVRF dialects and eventual end-user confusion.
- **Proof of Concept Container**: Support for inclusion of proof of concept / vulnerability reproduction steps. We have started to work on high-level plans to enable document producers to include step-by-step instructions to reproduce a vulnerability and include exploit code.

Special Thanks

I'd like to take a quick minute to call out to the CVRF Core Team who went above and beyond to ensure CVRF was built the way it needed to be:

- Joe Clarke, Cisco Systems
- Troy Fridley, Cisco Systems
- Joe Hemmerlein, Microsoft Corporation

Without your help we wouldn't be here today. Thanks guys.

