Writing an OVAL Definition

Version 5.3

DRAFT

Introduction **3**

Step 1 -- A New OVAL Definition XML File **3**

Step 2 -- Adding A Definition **4**

Assigning an ID *4*

The Version of a Definition *4*

What class does an OVAL Definition fall into? *4*

Step 3 -- Filling In The Metadata **5**

The Definition Title *5*

What about the <affected> element? *5*

Supplying References *5*

The Definition Description *5*

Appending Additional Metadata *5*

Step 4 -- Notes **5**

Step 5 -- Creating The Criteria **6**

Extending Other Definitions *6*

Step 6 -- Adding A Test **6**

Assigning an ID *7*

Check Existence *7*

Check *7*

Step 7 -- Adding Necessary Objects **7**

Step 8 -- Adding Necessary States **7**

Step 9 -- Submitting The OVAL Definition **7**

Introduction

To take advantage of the interoperability enabled by using a standard language like OVAL (Open Vulnerability and Assessment Language), someone has to generate the initial definitions to be used. An OVAL Definition is a bunch of expected system details arranged in a standard, pre-defined way, such that tool consuming this data know how to use it. An OVAL Definition can be written to describe a computer vulnerability, a policy that one must comply with, a system patch, or any other specific machine state.

Writing an OVAL Definition can be a tricky proposition for someone who is not familiar with XML. For those without XML experience, it is recommended to find an XML tool that will help with the process. This will help with things like validation of the definition against the OVAL Schema.

For those with XML experience, this document should help identify common problem areas and answer many of the questions one may have while attempting to write an OVAL Definition.

Step 1 -- A New OVAL Definition XML File

Before a new OVAL Definition can be written, the XML file must be set up. This XML file can be used to hold multiple new OVAL Definitions. The structure of this XML file is explained below.

The root element (the first XML tag, located at the top of the file) should be <oval_definitions>. The root element is also where each needed namespace is declared. Each component schema (independent, Windows, UNIX, Solairs, etc) used by the OVAL Definition being written must have its corresponding namespace declared in the root element. It is not uncommon to see 3 or 4 different namespaces declared. Note that the xsi namespace must also be declared (although you don't have to supply it or give a location for it) to support some of the xml constructs that are used.

In addition to namespaces, the root element also defines the location of the actual schema files needed for validation. There should be a location for each namespace declared.

Looking at the example below, each declared namespace is represented as xmlns:*identifier*. The xmlns attribute without an identifier is the default namespace.

```
<oval_definitions xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"
                  xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
                  xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5"
                  xmlns:ind-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"
                  xmlns:win-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://oval.mitre.org/XMLSchema/oval-common-5 oval-common-schema.xsd
                                      http://oval.mitre.org/XMLSchema/oval-definitions-5 oval-definitions-schema.xsd
                                      http://oval.mitre.org/XMLSchema/oval-definitions-5#independent independent-definitions-schema.xsd
                                      http://oval.mitre.org/XMLSchema/oval-definitions-5#windows windows-definitions-schema.xsd">
    <generator>
          <oval:product_name>Writer Tool</oval:product_name>
          <oval:product_version>2.3</oval:product_version>
          <oval:schema_version>5.3</oval:schema_version>
          <oval:timestamp>2007-10-12T18:13:45</oval:timestamp>
    </generator>

    ...

</oval_definitions>
```

Each XML file must also have a <generator> element that holds information about when the file was compiled and what version of the OVAL Schema was used. The optional <product_name> and <product_version> child elements describe the name and version of the application used to generate the file. The required <schema_version> child element holds the version of the OVAL Schema that the definition XML file has been written in and that should be used for validation. The required <timestamp> child element holds information about when the particular OVAL document was compiled. The format for the timestamp is yyyy-mm-ddThh:mm:ss.

Note that the information contained in the <generator> element does not specify when a definition (or set of definitions) was created or modified but rather when the actual XML file that contains the definition(s) was created. For example, the document might have pulled a bunch of existing OVAL Definitions together, each of the definitions having been created at some point in the past. The information in this case would be about when the combined document was created.

Step 2 -- Adding A Definition

Once the XML file has been set up, it is time to add an OVAL Definition to it. Remember that each definition should coincide with a complete statement about the state of a machine. Individual definitions are grouped together as children of the <definitions> element. Each individual definition has three required attributes: id, version, and class. These are described in more detail below. A definition is separated into three sections: a metadata section, a notes section, and a criteria section. See the following snippet for an example.

```
<oval_definitions>

    <generator>...</generator>

    <definitions>

          <definition id="oval:org.mitre.oval:def:1234" version="1" class="vulnerability">

                <metadata>...</metadata>
                <notes>...</notes>
                <criteria>...</criteria>

          </definition>

          <definition>...</definition>
          <definition>...</definition>
          <definition>...</definition>

    </definitions>

    <tests>...</tests>
    <objects>...</objects>
    <states>...</states>
    <variables>...</variables>

</oval_definitions>
```

Assigning an ID

For OVAL Definitions submitted to the OVAL Repository, they should use the org.mitre.oval id namespace. New ids are assigned randomly from a pool that is managed by the OVAL Repository. When submitting new content to the OVAL Repository all new items (definitions, test, objects, states, & variables) should be assigned temporary ids. Once a new submission is reviewed and imported into the OVAL Repository official ids will be assigned. Temporary ids should be created in a namespace other than the org.mitre.oval id namespace.

The Version of a Definition

The required version attribute holds the current version of the definition. Versions are integers, starting at 1 and incrementing every time a definition is modified.

What class does an OVAL Definition fall into?

Each definition must be assigned a *class* to help group the definition by the type of system state it is describing. This helps find and sort definitions when the need arises. There are 5 different classes to choose from: compliance, inventory, patch, vulnerability, and miscellaneous.

Compliance definitions describe the state of a machine when in compliance with a specific policy. Inventory definitions describe whether a specific piece of software is installed on the system. Patch definitions are used to describe the conditions under which a patch executable should be installed. Vulnerability definitions are used to describe the condition under which a vulnerability exists on a system. Finally, the miscellaneous class is used to categorize a definition that doesn't fit into one of the other four classes.

Step 3 -- Filling In The Metadata

Each OVAL Definition should have some metadata associated with it to help identify it amongst a large collection of definitions.

```
<metadata>
      <title>...</title>
      <affected>...</affected>
      <reference>...</reference>
      <description>...</description>
</metadata>
```

The Definition Title

The required title child element holds a short string that is used to quickly identify the definition to a human user.

What about the <affected> element?

The affected metadata item contains information about the system(s) for which the definition has been written. Remember that this is just metadata and not part of the criteria.

Supplying References

References can be added to the metadata section of an OVAL Definition to link it to a definitive external name. For example, a vulnerability definition can include a reference to the CVE Identifier that represent the vulnerability being defined. Each reference element found in an OVAL Definition can include the external ID (for example the CVE ID) and a URL for finding information about the external ID.

```
<reference source="CVE" ref_id="CAN-2005-0063" ref_url="http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-0063"/>
```

The Definition Description

The description of the OVAL definition will contain the description of the vulnerability contained in the CVE entry. In the case of vulnerability definitions without a CVE identifier, compliance or patch definitions the description should adequately convey the purpose of the definition. It should either describe the vulnerability in question, the patch and its function or what constitutes compliance. This section is for human consumption so it is left to the definition writer's discretion to determine relevant content.

Appending Additional Metadata

Additional metadata is also allowed although it is not part of the official OVAL Schema. Individual organizations can place metadata items that they feel are important and these will be skipped during the validation. All OVAL really cares about is that the stated metadata items are there.

Step 4 -- Notes

Each definition can have optional notes associated with it. Each note contains some information about the definition or about the tests that the definition references. A note may record an unresolved question about the definition or test or present the reason as to why a particular approach was taken. To add a note(s), simply add a <note> element as a child of the <notes> element. See the example below.

```
<notes>
    <note>This could be a note about an approach taken or why certain tests were included in the definition.</note>
    <note>This note might be about a resource that was used.</note>
</notes>
```

Step 5 -- Creating The Criteria

Once the metadata and any notes have been added to the definition, it is time to start building the criteria. The criteria portion of an OVAL Definition is the meat of what is actually being described. It joins individual tests together with a logical operator to specify the specific computer state.

To get started, a <criteria> element is added as a child of the <definition> element. Inside each <criteria> element can be found other <criteria> elements, <criterion> elements, and <extended_definition> elements. See the example below regarding this relationship.

```
<criteria operator="OR">

    <criteria operator="AND">
          <criterion test_ref="oval:org.mitre.oval:tst:123" comment="Windows XP is installed"/>
          <criterion test_ref="oval:org.mitre.oval:tst:234" comment="file foo.txt exists"/>
    </criteria>

    <criteria operator="AND" negate="true">
          <criterion test_ref="oval:org.mitre.oval:tst:345" comment="Windows 2003 is installed"/>
          <criterion test_ref="oval:org.mitre.oval:tst:456" comment="file fred.txt has a version less than 2"/>
          <criterion test_ref="oval:org.mitre.oval:tst:567" negate="true" comment=patch is installed"/>
    </criteria>

</criteria>
```

Each <criteria> element has an operator attribute who's value is either AND, OR, XOR, or ONE. This value determines how the result of the criteria statement will be determined. If the operator is AND, then every test referenced by the criteria must be true for the entire criteria to return true. If the operator is OR, then at least one test referenced by the criteria must be true for the entire criteria to return true.

In addition to setting an operator, the result of the criteria can be negated. This means that if the result is generated by combining the operator with the test references is true, then the negation changes it to false. The opposite is true if the generated result of the test references is false as the negation changes it to true.

As mentioned before, the criteria is a recursive element. That means you can have new <criteria> elements inside of existing <criteria> elements. The reason for this is to enable nesting of logical statements. The ability to nest these criteria statements allows complex logic to be

Extending Other Definitions

The optional extend_definition element allows existing definitions to be included in the criteria. This works by evaluating the extended definition and then using the result within the logical context of the extending definition.

Step 6 -- Adding A Test

Once the criteria of the OVAL Definition has been created, the next step is to add the associated tests.

```
<oval_definitions>

    <generator>...</generator>

    <definitions>

          <definition>...</definition>
          <definition>...</definition>
          <definition>...</definition>

    </definitions>

    <tests>

          <file_test id="oval:org.mitre.oval:tst:5678" version="1" check_existence="" check="" comment="">
                <object object_ref="">
                <state state_ref=""/>
          </file_test>

    </tests>

    <objects>...</objects>
    <states>...</states>
    <variables>...</variables>
```

```
</oval_definitions>
```

Assigning an ID

Check Existence

The optional check_existence attribute determines how many objects in the specified set must exists for the test to be true. For example, if a value of 'all_exist' is given, every object defined by the OVAL Object must exist on the system for the test to return true. If the OVAL Object uses a variable reference, then every value of that variable must exist. Note that a pattern match defines a set of matching objects found on a system. So when check_existence = 'all_exist' and a regex matches anything on a system the test will evaluate to true. (since all matching objects on the system were found on the system) When check_existence = 'all_exist' and a regex does not match anything on a system the test will evaluate to false.

Check

The required check attribute determines how many of the existing objects must satisfy the state requirements. (For example: Should the test check that all files match a specified version or that at least one file matches the specified version?) Note that if the test does not contain any references to OVAL States, then the check attribute has no meaning and can be ignored during evaluation.

Step 7 -- Adding Necessary Objects

Step 8 -- Adding Necessary States

Step 9 -- Submitting The OVAL Definition

If desired, submissions of OVAL vulnerability, compliance, inventory, and patch definitions may be emailed to MITRE in the proper format by members of the security community who have registered for the Community Forum, or by OVAL Board members. All submissions will be reviewed by the OVAL content team and the OVAL Editor prior to being posted on the OVAL Web site for comment and public debate.

To submit an OVAL definition:

1. Review the official OVAL Definition Schema.
2. Follow the instructions for writing the definition per the OVAL definition Submission Guidelines below.
3. Community Forum members should email the draft to oval-discussion-list@lists.mitre.org.

Those wishing to submit sensitive information may send it directly to oval@mitre.org.