折页册

品质及数值

# 这份资讯应该要清楚

当读者打开折页册时，这将是他们最先看到的文字，此处是简短扼要地摘要您所提供之产品或服务的好位置。

## 品质及数值

### 这份资讯应该要清楚

# 这份资讯应该要清楚

当读者打开折页册时，这将是他们最先看到的文字，此处是简短扼要地摘要您所提供之产品或服务的好位置。

[OVAL Docs](http://ovalproject.github.io/)

* [Getting Started](http://ovalproject.github.io/getting-started/tutorial/)
* [Documentation](http://ovalproject.github.io/getting-started/tutorial/)
* [OVAL Board](http://ovalproject.github.io/getting-started/tutorial/)

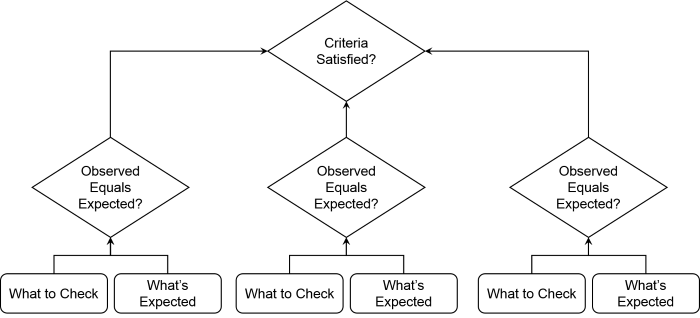
# **OVAL Content Creation Tutorial**

## **Introduction**

The Open Vulnerability and Assessment Language (OVAL) is an XML-based community standard for representing and exchanging security content. Its purpose is to enable the transfer of information across the entire spectrum of security tools and services. OVAL provides a standardized representation of the supporting data needed for the three main steps of the security assessment process: OVAL Definitions for representing the expected state of an endpoint, OVAL System Characteristics for representing the actual state of an endpoint, and OVAL Results for reporting the outcome of the assessment. More information about how OVAL supports the steps of the security assessment process can be found at: http://oval.mitre.org/language/about/overview.html#how\_oval\_works. This guide addresses the development of OVAL Definitions, often referred to as content creation. It explains the structure and components of an OVAL Definition as well as provides an overview of the process for creating definitions and related best practices. Also covered are the resources, tools, and methods available for generating OVAL Definitions.

## **General Concepts**

OVAL Definitions provide a means to specify what endpoint information should be checked and what corresponding values are expected to be found. In addition, an OVAL Definition defines how to interpret the results of comparing the characteristics, which were observed on an endpoint, against what was expected. Figure-1, below, depicts a generic flowchart of this process.

 **Figure-1 Comparing and Interpreting Endpoint Information**

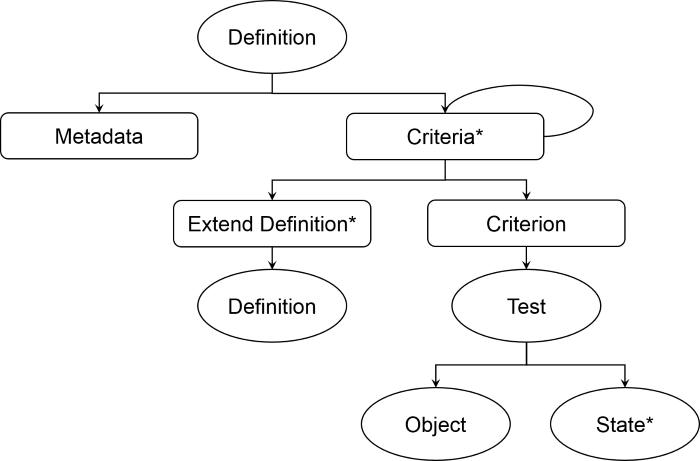
In Figure-1, a number of tests are defined in terms of what endpoint information should be checked and what information is expected to be found on the endpoint. Given these kinds of tests, an automated process can be performed which gathers the actual endpoint information and makes the comparisons.  
Furthermore, criteria are defined that can be automatically assessed, based on the outcomes of the tests.

The components of an OVAL Definition follow the same pattern as the general concepts discussed above. The components relate to Figure-1 in the following fashion.

* An OVAL Object specifies “What to Check” on the endpoint being evaluated.
* OVAL States specify “What’s Expected” in terms of the information found concerning the endpoint.
* The OVAL Test associates OVAL Objects and OVAL States which should be used to determine whether the “Observed Equals Expected”.
* Finally, the OVAL Criteria describes an assertion about an endpoint which is used to determine whether the “Criteria Is Satisfied.” The OVAL Criteria defines a logical expression used to interpret the outcome of the comparisons specified by the OVAL Tests.

## **OVAL Definition Components and Structure**

Figure-2, below, documents the OVAL Definition in greater detail. Rectangles in the figure represent properties of the definition. The circular shapes represent other OVAL components with which a Definition may be associated. The lines in the diagram represent the relationships among these components. In addition, optional components in the diagram are marked with an asterisk (‘\*’).

 **Figure-2 OVAL Definition Components and Structure**

As seen in Figure-2, an OVAL Definition includes metadata which describes the purpose and origin of the definition, in addition to OVAL Criteria. The OVAL Criteria is one of the building blocks for assembling the assertion which the definition is designed to evaluate. The OVAL Criteria and OVAL Criterion are used together to create a logical statement which references OVAL Tests and other OVAL Definitions.  
(Other definitions are referenced via the extend definition component.) As noted before, an OVAL Test associates OVAL Objects and OVAL States to check specific information on the endpoint. Each OVAL Test provides a Boolean result used in evaluating the logical statement formed by the OVAL Criteria and Criterion.

The OVAL Language is expressed as XML. The XML format is defined in several XML Schemas. Further information concerning these schemas can be found in the OVAL Language Specification (http://oval.mitre.org/language/version5.10.1/#specification). This section introduces XML examples of OVAL Language components.

### **Metadata**

The metadata element in an OVAL Definition conveys information about the definition. This includes a definition title, the operating systems and platforms the definition applies to, and a description of what the definition is checking for. Note that information in the metadata element, including platforms and products, does not affect evaluation of the definition.

<metadata>

<title>CoolWare NET-Suite is installed on the endpoint</title>

<affected family=”windows”>

<platform>Microsoft Windows 98</platform>

<platform>Microsoft Windows 2000</platform>

<platform>Microsoft Windows XP</platform>

<product>CoolWare Net-Suite</product>

</affected>

<description>CoolWare NET-Suite is installed</description>

</metadata>

### **Object**

An OVAL Object specifies which information should be collected from the endpoint for evaluation. An OVAL Object must provide sufficient entities for a user to uniquely identify the endpoint information to be collected. In the example below, the OVAL Object specifies that a key in the Windows registry, which contains version information about an application called CoolWare iBrowse, should be collected from the endpoint.

<registry\_object id=”oval:tutorial:obj:1” version=”3”

comment=”The registry key which holds the version of CoolWare iBrowse”

xmlns=”http://oval.mitre.org/XMLSchema/oval-definitions-5#windows”>

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SOFTWARE\CoolWare\iBrowse</key>

<name>Version</name>

</registry\_object>

### **State**

An OVAL State describes the expected values which are compared to the information collected from the endpoint being evaluated. In the example below, the registry\_state specifies that information matching the value, “1.0”, is expected to be found in the Windows registry.

<registry\_state id=”oval:tutorial:ste:1” version=”2”

comment=”The registry key matches with CoolWare iBrowse version 1.0

installed”

xmlns=”http://oval.mitre.org/XMLSchema/oval-definitions-5#windows”>

<value>1.0</value>

</registry\_state>

### **Test**

An OVAL Test defines the relationship between an OVAL Object and zero or more OVAL States. It matches the Definition of the endpoint information to be collected from the endpoint with the corresponding values expected to be found. In the example registry\_test below, the OVAL Object, “oval:tutorial:obj:1” is associated with the OVAL State, “oval:tutorial:ste:1”.

<registry\_test id=”oval:tutorial:tst:1” version=”4”

comment=”CoolWare iBrowse version 1.0 is installed”

check\_existence=”at\_least\_one\_exists” check=”all”

xmlns=”http://oval.mitre.org/XMLSchema/oval-definitions-5#windows”>

<object object\_ref=”oval:tutorial:obj:1”/>

<state state\_ref=”oval:tutorial:ste:1”/>

</registry\_test>

The “check” and “check\_existence” attributes in an OVAL Test are used to guide the comparison of endpoint values. The check\_existence attribute defines how many distinct groupings of information, as defined by the OVAL Object, must exist on the endpoint for the OVAL Test to evaluate to ‘true’. The check attribute defines how many of the collected values must satisfy the requirements given in the OVAL State for the OVAL Test to evaluate to ‘true’. In the example above, the check\_existence property indicates that at least one instance of the information identified by oval:tutorial:obj:1 must be found on the endpoint (i.e., “at\_least\_one\_exists”) and that all values of the information, specified by oval:tutorial:ste:1, must be checked against information found on the endpoint (i.e., “all”).

### **Criteria/Criterion**

The OVAL Criteria defines the logical expression in an OVAL Definition, and may contain zero or more OVAL Criterion and nested Criteria. The OVAL Criterion references OVAL Tests and represents a term in the logical expression defined by the OVAL Criteria. In the example below, the OVAL Criteria contains two OVAL Criterion. The first OVAL Criterion checks whether CoolWare iBrowse is installed. The second OVAL Criterion checks whether CoolWare eMail is installed. So, the logical expression defined by the OVAL Criteria below checks whether both iBrowse and eMail are installed on the endpoint being evaluated.

<criteria>

<criterion comment="CoolWare iBrowse version 1.0 is installed"

test\_ref="oval:tutorial:tst:1"/>

<criterion comment="CoolWare eMail version 1.5 is installed"

test\_ref="oval:tutorial:tst:2"/>

</criteria>

### **Definition**

In addition to OVAL Metadata and OVAL Criteria (as illustrated in Figure-2, above), an OVAL Definition also has a class which indicates the category the definition falls into. This helps to identify the definition’s purpose. In OVAL, there are five kinds of definition classes:

1. Compliance – Checks whether an endpoint is compliant with a specific policy.
2. Inventory – Checks whether specific software is installed on the endpoint.
3. Miscellaneous – OVAL Definitions that do not fall into one of the other defined classes.
4. Patch – Checks whether a patch needs to be installed on an endpoint.
5. Vulnerability – Checks whether an endpoint is vulnerable.

The definition below has been constructed from some of the example components discussed above.  
These components have been placed within a definition element. Note that this definition has a class of “inventory”, since it is checking to determine whether particular software is installed on the endpoint. The definition is checking for CoolWare’s Net-Suite, which is indicated by both iBrowse and eMail being installed on the endpoint.

<definition id="oval:tutorial:def:123" version="1" class="inventory">

<metadata>

<title>CoolWare NET-Suite is installed on the endpoint</title>

<affected family="windows">

<platform>Microsoft Windows 98</platform>

<platform>Microsoft Windows 2000</platform>

<platform>Microsoft Windows XP</platform>

<product>CoolWare Net-Suite</product>

</affected>

<description>CoolWare NET-Suite is installed</description>

</metadata>

<criteria>

<criterion comment="CoolWare iBrowse version 1.0 is installed"

test\_ref="oval:tutorial:tst:1"/>

<criterion comment="CoolWare eMail version 1.5 is installed"

test\_ref="oval:tutorial:tst:2"/>

</criteria>

</definition>

### **Variables**

OVAL Variables provide the means to define a grouping of one of more values which may be referenced within other OVAL content. For example, consider the registry\_state below. It references an OVAL Variable to define what values of a registry key to check for. In addition to specifying the OVAL Variable, the OVAL State must also stipulate what datatype and operation should be applied to the values provided by the OVAL Variable.

<registry\_state id="oval:tutorial:ste:2" version="3"

comment="The registry key matches CoolWare Products specified below"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<value datatype="string" operation ="equals”

var\_ref="oval:tutorial: var:1" var\_check=”all/>

</registry\_state>

The referenced OVAL Variable is shown below. It is composed of a list of product names. Through the variable, each of the product names is referenced by the OVAL State above.

<constant\_variable id="oval:tutorial:var:1" version="1"

datatype="string"

comment="Specific CoolWare products to check for">

<value>iBrowse</value>

<value>eMail</value>

<value>Cool Graphs</value>

<value>Einstein Math Editor</value>

</constant\_variable>

Note that there are three kinds of variables in the OVAL language. In this case, a “constant\_variable” which defines literal values is utilized. The OVAL Language also provides local and external variables.  
These are discussed in the OVAL Language Specification.

### **Sets**

The OVAL Set construct provides a way to express complex OVAL Objects which are the result of logically combining other OVAL Objects. Below, an OVAL Set is created to combine two other objects using the union operator. For example, if the OVAL Objects, “oval:tutorial:obj:33” and “oval:tutorial:obj:44” were file\_objects, then “oval:tutorial:obj:55” would identify all the files specified by both prior OVAL Objects.

<file\_object id="oval:tutorial:obj:55" version=”1”

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

<oval-def:set set\_operator=”UNION”>

<oval-def:object\_reference>oval:tutorial:obj:33

</oval-def:object\_reference>

<oval-def:object\_reference>oval:tutorial:obj:44

</oval-def:object\_reference>

</oval-def:set>

</file\_object>

### **Filters**

The OVAL Filter construct allows the explicit inclusion or exclusion of specific information from a grouping of endpoint information, based on an OVAL State. In the example below, the file\_state, oval:tutorial:ste:55, which will be referenced in the filter, identifies files which are owned by the user, “755”.

<file\_state id="oval:tutorial:ste:55" version="1"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

<user\_id operation="equals" datatype="int">755</user\_id>

</file\_state>

Below, an OVAL Filter is used to constrain an OVAL file\_object. It does this by referencing the OVAL State, oval:tutorial:ste:55, as defined above. Since the OVAL Filter references this OVAL State, only the files owned by the user, with user ID (UID) “755”, would be included. All other UIDs would be filtered out.

<file\_object id="oval:tutorial:obj:66" " version=”1”

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

<path operation="pattern match">.\*</path>

<filename operation="pattern match">.\*</filename>

<oval-def:filter action="include">oval:tutorial:ste:55

</oval-def:filter>

</file\_object>

### **Regular Expressions**

The OVAL Language supports a common subset of the regular expression character classes, operations, expressions, and other lexical tokens defined in Perl 5’s regular expression specification. More details on regular expression support can be found in Appendix D of the OVAL Language Specification. One purpose of regular expressions in OVAL is to increase the flexibility of OVAL Definitions. In the example below, a regular expression is used in an OVAL State to represent all premium versions of CoolWare products which may be installed on the endpoint.

First, consider the registry\_object, “oval:tutorial:obj:1. It specifies that the registry key which stores the names of CoolWare products should be checked.

<registry\_object id="oval:tutorial:obj:1" version="3"

comment="The registry key which holds the names of CoolWare products"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SOFTWARE\CoolWare</key>

<name>Product</name>

</registry\_object>

The registry\_state below uses a regular expression to define what values are expected. When referenced from within an OVAL Test, in combination with the registry\_object above, this registry\_state defines the names of all premium versions of CoolWare products. The regular expression is crafted to find all product names ending with the word “Premium” since this is how premium versions of CoolWare products are indicated.

<registry\_state id="oval:tutorial:ste:1" version="2"

comment="The registry key that matches Premium editions of a CoolWare

product"

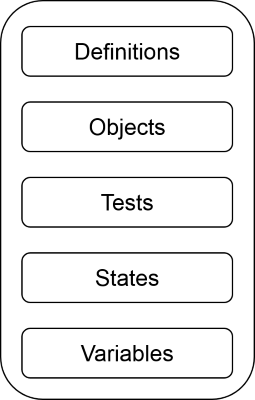
xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<value datatype=”string” operation="pattern match”>.\*Premium$</value>

</registry\_state>

## **OVAL Definitions Document**

OVAL Definitions and their required components are documented and exchanged in XML as children of the “oval\_definitions” element. The oval\_definitions element includes the OVAL Definitions to be exchanged, along with the OVAL Tests, OVAL Objects and OVAL States each OVAL Definition references in its specification. Figure-3 illustrates the component sections included in the OVAL Definitions element, in the order in which they occur. In addition to the components that have already been discussed, the OVAL oval\_definitions element also contains a section for OVAL Variables. OVAL Variables will be covered later in this document.



**Figure-3 oval\_definitions Element Sections**

The example oval\_definitions element, below, includes the definition discussed above, and all other OVAL components required for specifying it. The definition and other components which have already been discussed in this document are included in the example. An additional OVAL Test, OVAL Object, and OVAL State, which have not been covered yet, are also included. Since these components are required for the example definition they must also be included in the oval\_definitions element.

**<?xml version="1.0" encoding="UTF-8"?>**

<oval\_definitions

xsi:schemaLocation="http://oval.mitre.org/XMLSchema/oval-definitions-5

oval-definitions-schema.xsd http://oval.mitre.org/XMLSchema/oval-definitions-5#windows

windows-definitions-schema.xsd http://oval.mitre.org/XMLSchema/oval-common-5

oval-common-schema.xsd"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"

xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">

<generator>

<oval:product\_name>OVAL-office</oval:product\_name>

<oval:schema\_version>5.10.1</oval:schema\_version>

<oval:timestamp>2014-10-09T14:11:39.105-04:00</oval:timestamp>

</generator>

<definitions>

<definition id="oval:tutorial:def:123" version="1" class="inventory">

<metadata>

<title>CoolWare NET-Suite is installed on the endpoint</title>

<affected family="windows">

<platform>Microsoft Windows 98</platform>

<platform>Microsoft Windows 2000</platform>

<platform>Microsoft Windows XP</platform>

<product>CoolWare Net-Suite</product>

</affected>

<description>CoolWare NET-Suite is installed</description>

</metadata>

<criteria>

<criterion comment="CoolWare iBrowse version 1.0 is installed"

test\_ref="oval:tutorial:tst:1"/>

<criterion comment="CoolWare eMail version 1.5 is installed"

test\_ref="oval:tutorial:tst:2"/>

</criteria>

</definition>

</definitions>

<tests>

<registry\_test id="oval:tutorial:tst:1" version="4"

comment="CoolWare iBrowse version 1.0 is installed"

check\_existence="at\_least\_one\_exists" check="all"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<object object\_ref="oval:tutorial:obj:1"/>

<state state\_ref="oval:tutorial:ste:1"/>

</registry\_test>

<registry\_test id="oval:tutorial:tst:2" version="2"

comment="CoolWare eMail version 1.5 is installed"

check\_existence="at\_least\_one\_exists" check="all"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<object object\_ref="oval:tutorial:obj:2"/>

<state state\_ref="oval:tutorial:ste:2"/>

</registry\_test>

</tests>

<objects>

<registry\_object id="oval:tutorial:obj:1" version="3"

comment="The registry key which holds the version of CoolWare iBrowse"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SOFTWARE\CoolWare\iBrowse</key>

<name>Version</version>

</registry\_object>

<registry\_object id="oval:tutorial:obj:2" version="1"

comment="The registry key which holds the version of CoolWare eMail"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SOFTWARE\CoolWare\eMail</key>

<name>Version</name>

</registry\_object>

</objects>

<states>

<registry\_state id="oval:tutorial:ste:1" version="2"

comment="The registry key matches with CoolWare iBrowse version 1.0 installed"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<value>1.0</value>

</registry\_state>

<registry\_state id="oval:tutorial:ste:2" version="3"

comment="The registry key matches with CoolWare eMail installed version"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<value>1.5</value>

</registry\_state>

</states>

</oval\_definitions>

## **Authoring Definition Content**

Producing OVAL Definitions is the process by which information, from an external source, is transformed into an OVAL Definition. Often the source of the information is a security advisory, configuration checklist, or other data feed. Other times, this information must be created through detailed endpoint investigation and research of known issues. In either case, endpoint information is encoded in the form of an assertion. This section discusses the definition development process mainly from a manual perspective, but is also applicable to automation.

In developing an OVAL Definition, a combination of research and acquisition of existing information may be necessary. It is also likely that the process of developing definition components will be iterative. For instance, it may not be possible to completely define the assertion before investigating what information is available from the endpoint. In addition, existing OVAL content should be reused whenever possible. (The OVAL Language enables content reuse through the use of globally unique IDs for OVAL Definitions, OVAL Tests, OVAL Objects, OVAL States, and OVAL Variables.) These topics and related considerations are discussed in more detail, below.

### **Required Tools and Skills**

To develop an OVAL Definition, an author should have some required knowledge and access to basic tools. The author should also have domain expertise in the platform(s) that will be evaluated. If the definition is based on existing information, an ability to interpret guidance written in prose is also needed as documents like security bulletins are often conveyed using informal language. Since OVAL is an XML-based language, basic XML tools and skills will also be required. The definition author should be able to read and understand XML documents and schema. In addition, an ability to operate XML authoring and validation tools will be required to promote correct construction of the OVAL Definition.

The OVAL Language makes use of regular expressions to specify some aspects of an OVAL Definition. So the author needs the ability to write and understand regular expressions. For automation purposes, the ability to write and understand XPath statements will also be needed, as well as the ability to write programs/scripts to generate definition content.

A basic understanding of Boolean algebra is also required for content creation. This is because the OVAL Language uses logical expressions to define the combination of test outcomes which are expected. For example, the criteria element can express that outcomes should be combined with AND/OR operations to evaluate the configuration of an endpoint. Another example is that test outcomes can be negated using the Boolean NOT operator to express that the test is expected to fail.

### **Acquiring Existing Information**

In many cases, the necessary information about an endpoint and how to detect it has already been investigated. This information is available in different forms, such as security advisories and configuration policies. The source may describe how to discover whether or not a given patch should be installed on an endpoint, or could be prescriptive as in the case of an approved software inventory. When developing an OVAL Definition from existing information, the source should be authoritative. It is usually best to base definitions on information released by primary system vendors. Regardless of where the existing information was obtained, the source should be documented in the definition’s metadata property.

### **Researching OVAL Content**

When existing information is not available or is incomplete, the definition author will need to perform research in order to develop definition content. In the course of the research, a number of questions are pertinent.

* What is the definition going to assert/check?
* What platforms will the definition target?
* What existing OVAL content can be reused?
* What data must be collected about the endpoint to evaluate the assertion?
* How will the endpoint data be identified?
* What are the logical relationships required for the assertion?

Note that the questions discussed below may not be answerable in a single, linear pass. Some questions may need to be revisited resulting in an iterative development process. Note also that the questions discussed are not meant to be an exhaustive checklist but as a guide. Additional questions may be appropriate, depending on your development process.

The first questions to address pertain to the intended scope of the definition. What is the definition intended to assert (or check)? This question indicates the ultimate purpose of the definition, and should be thought through. Another question to address early in definition development concerns the platforms to which the definition will be applicable. This question helps to focus the definition development. (Note also that these platforms are documented in the definition metadata.)

Throughout the development process, it is important to identify what existing OVAL content might be reused in developing the definition. Some of the necessary research and development may have already been done by the community, and there may be existing OVAL Definitions, OVAL Objects, etc., which can be leveraged. In addition to the OVAL Repository maintained by MITRE on the behalf of DHS (see http://oval.mitre.org/repository/), other OVAL repositories are available (see http://oval.mitre.org/repository/about/other\_repositories.html) to consult.

The next questions are related to the formal construction of the assertion about the endpoint(s) addressed by the definition. What data must be collected about the endpoint(s) to support the assertion and how is this data identified on the endpoint? What are the logical relationships necessary to express the assertion? The answers to these questions will be refined to develop definition components such as OVAL Objects, OVAL States, OVAL Tests, and OVAL Criteria. This is discussed in the next section.

### **Expressing OVAL Content**

The central task in developing an OVAL Definition is to flesh out its assertion. This assertion is a logical statement built with OVAL Criteria, Criterion and extended definitions. These components reference OVAL Tests and use Boolean operators to define the assertion to be evaluated. The referenced OVAL Test components define how the true/false values used to evaluate the OVAL Criteria are generated.

OVAL Tests match the identified endpoint information with the corresponding values desired to be found on the endpoint. These comparisons reduce to true/false values which are used to evaluate the logical statement described above. Based on the platform(s) and the assertion, the appropriate OVAL Tests are configured to support the evaluation of the assertion specified in the definition.

The OVAL Object component provides the means to identify the information to be collected from the actual endpoint under consideration. The desired values to be found on the endpoint are represented by OVAL States. The OVAL Test matches the identified endpoint values (OVAL Objects) with their desired values (OVAL States) to define comparisons which evaluate to the true/false values referenced by OVAL Criterion.

### **Populating Metadata**

As discussed previously, an OVAL Definition includes a metadata property. Accurate and complete metadata is important for describing the purpose and scope of the definition. Best practices for populating metadata properties are discussed below.

The platform and product properties of an OVAL Definition’s metadata property are used to provide a listing of platforms and products to which the OVAL Definition is known to apply. It is important that these properties completely describe the applicability of the definition, both to ensure correct application of the definition and to facilitate reuse. It should also be noted that the values of the platform and product properties do not impact the evaluation of the definition. For example, it is not required that the platform being evaluated be documented in the platform property. The definition can be applied to a platform whether or not it is documented in the definition metadata.

The reference property of an OVAL Definition’s metadata property is used provide an authoritative citation for the specific endpoint information being described by the OVAL Definition.

### **Testing**

Testing is an important part of the OVAL content development process. The aim of content testing is to evaluate the following aspects of an OVAL Definition:

* Validity. The definition must be conformant to the OVAL Language specification. In other words, it must be valid OVAL content.
* Interoperability. The definition must be interoperable across different implementations of interpreters and other OVAL tools.
* Accuracy. The definition must make accurate evaluations of the endpoints it was written to address.
* Efficiency. It must be computationally practical to evaluate an endpoint using the definition.

These aspects are discussed in greater detail, below.

The definition must be valid OVAL content. Checking the validity of OVAL content is supported by automated XML tools. The OVAL language is described by a suite of XML Schema and Schematron documents. An XML validating parser is required to check conformance to the XML Schemas developed for OVAL. The XML Schema specification is a W3C recommendation and a number of tools (both open source and commercially developed) are available for XML Schema validation.

XML Schematron rules have also been made available to support OVAL content validation. Schematron is able to define constraints not expressible in XML Schema and is used to further refine the definition of the OVAL language. Schematron is an ISO standard and validation tools are available to check OVAL content against Schematron rules.

In addition to constraints formally defined in XML Schema and Schematron, there are restrictions applicable to OVAL content which are not expressed in an automatable format. An example is the format of IP addresses. These are not formally defined in XML Schema or Schematron, but are described by prose in the OVAL Language Specification. It is required that these kinds of constraints are also enforced. So an OVAL Definition may be valid with respect to the XML Schema and Schematron definitions, but may not be valid OVAL.

In addition to validity, aspects concerning the runtime application of the definition must also be considered. At a minimum, the definition should be evaluated by more than one OVAL interpreter to check for potential interoperability issues. Ideally, the definition should be tested on all the OVAL interpreters/tools to which the definition author has access.

Running the definition on a real interpreter to evaluate a real endpoint also provides the opportunity to test the accuracy of the definition. It is important to check for false positives and negatives which may be produced by the OVAL Criteria. This could indicate erroneous or incomplete research in developing the definition or simply an error in formulating the OVAL Criteria. In either case, there should be a high level of confidence established before the definition is used in a production environment.

Finally, the efficiency of the definition should also be assessed. A definition may be valid, interoperable, and accurate but still not use-able in practice.

## **Annotated Examples**

This section includes common examples of OVAL Content. Each example has been annotated with embedded comments to explain OVAL concepts and specific uses of OVAL components. The examples are listed below, along with a brief description of their purpose and the OVAL components they utilize.

* Checking for World Writeable Files. This OVAL definition example checks whether there are any world-writeable files on the endpoint. It uses the OVAL file\_test to determine permissions associated with the files.
* Inventory Example. This OVAL definition example checks for a bundled product. If the correct versions of the bundled software applications are installed, then the bundled product has been installed. The example uses the OVAL registry\_test to determine what software is installed, an OVAL Variable to list the expected applications, and a regular expression to check versions.
* Retrieving a File Path from an OVAL Registry Object. This OVAL Test example uses a local variable to determine the path used in a file\_object. The local variable is populated from information found in the Windows registry, and is retrieved using a registry\_object.
* Patch Example. This OVAL definition example checks whether a service pack is installed on the endpoint. It uses the OVAL registy\_test to check software versions to determine patch status.  
  This example also illustrates the use of OVAL extended definitions.
* Checking for Compliance to a Whitelist. This OVAL definition example checks installed software against an approved whitelist. If any software that is not on the whitelist is found, then the endpoint is not in compliance. This example uses an OVAL Variable to represent the software whitelist and demonstrates the use of OVAL Filters.

### **Checking for World Writable Files**

**<?xml version="1.0" encoding="UTF-8"?>**

*<!--*

*- World Writable Files Example -*

*This OVAL example checks whether there are any*

*world-writable files on the endpoint*

*This example uses the OVAL file\_test to determine permissions*

*associated with files on the endpoint.*

*-->*

*<!--*

*The oval\_definitions element is always the root and contains*

*the OVAL Content to be exchanged.*

*Schemas and namespaces needed for OVAL Definitions relevant to*

*UNIX are referenced as attributes in this oval\_defintions element*

*because OVAL components for UNIX are required to specify this*

*OVAL Content.*

*-->*

<oval\_definitions

xsi:schemaLocation="

http://oval.mitre.org/XMLSchema/oval-definitions-5 oval-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-common-5 oval-common-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#unix unix-definitions-schema.xsd"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"

xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">

*<!--*

*The generator element provides metadata about the tool/application*

*used to develop the OVAL Content.*

*-->*

<generator>

<oval:product\_name>OVAL-Office</oval:product\_name>

<oval:schema\_version>5.10.1</oval:schema\_version>

<oval:timestamp>2014-11-17T13:22:48.169-05:00</oval:timestamp>

</generator>

<tests>

*<!--*

*The tests element contains the OVAL Test(s). OVAL Tests specify what to*

*search for on an endpoint and what is expected to be found. The OVAL file\_test*

*is used to check for the existence of files on the endpoint.*

*-->*

<file\_test id="oval:tutorial:tst:1" version="1"

comment="Check that there are no world writable files" check="all"

check\_existence="none\_exist"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

*<!--*

*This file\_test searches for the files described by "oval:tutorial:obj:1.*

*-->*

<object object\_ref="oval:tutorial:obj:1"/>

</file\_test>

</tests>

<objects>

*<!--*

*The objects element contains the OVAL Object(s). The OVAL file\_object describes*

*what files to search for on the endpoint.*

*-->*

<file\_object id="oval:tutorial:obj:1" version="1"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

*<!--*

*This file object checks all the files on the system, but uses an OVAL filter to*

*include only the files described by "oval:tutorial:ste:1"*

*-->*

<path operation="pattern match">.\*</path>

<filename operation="pattern match">.\*</filename>

<oval-def:filter action="include">oval:tutorial:ste:1</oval-def:filter>

</file\_object>

</objects>

<states>

*<!--*

*The states element contains the OVAL State(s). The OVAL file\_state describes the files*

*expected to be found on the endpoint.*

*-->*

<file\_state id="oval:tutorial:ste:1" version="1"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

*<!--*

*This file\_state describes all files on the endpoint which are world-writable.*

*-->*

<owrite operation="equals" datatype="boolean">true</owrite>

</file\_state>

</states>

</oval\_definitions>

### **Inventory Example**

**<?xml version="1.0" encoding="UTF-8"?>**

*<!--*

*- Inventory Example using a regular expression and a variable -*

*This OVAL definition example checks whether the CoolWare*

*NET-Suite product has been installed on the endpoint. CoolWare*

*NET-Suite consists of particular versions of iBrowse and*

*any versions of other specified CoolWare products. If the*

*correct version of CoolWare iBrowse is currently installed on the*

*endpoint, and the specified CoolWare products are also installed,*

*then it can be concluded that NET-Suite has been installed.*

*This example uses the OVAL registry\_test to determine what software*

*is installed. A regular expression is used to check for the version*

*of iBrowse, and a variable is used to list the products included*

*in NET-Suite.*

*-->*

*<!--*

*The oval\_definitions element is always the root and contains*

*the OVAL Content to be exchanged.*

*Schemas and namespaces needed for OVAL Definitions relevant to*

*Windows are referenced as attributes in this oval\_defintions element*

*because OVAL components for Windows are required to specify this*

*OVAL Content.*

*-->*

<oval\_definitions

xsi:schemaLocation="

http://oval.mitre.org/XMLSchema/oval-definitions-5

oval-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#windows

windows-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-common-5

oval-common-schema.xsd"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"

xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">

<generator>

*<!--*

*The generator element provides metadata about the tool/application*

*used to develop the OVAL Content.*

*-->*

<oval:product\_name>OVAL-office</oval:product\_name>

<oval:schema\_version>5.10.1</oval:schema\_version>

<oval:timestamp>2014-10-09T14:11:39.105-04:00</oval:timestamp>

</generator>

<definitions>

*<!--*

*The definitions element contains the OVAL definition(s) to be exchanged.*

*-->*

<definition id="oval:tutorial:def:123" version="1" class="inventory">

*<!--*

*This definition checks software inventory.*

*-->*

<metadata>

*<!--*

*The metadata element contains information about the definition,*

*including its title and description. This definition checks*

*whether the product called CoolWare NET-Suite is installed.*

*-->*

<title>CoolWare NET-Suite is installed</title>

<affected family="windows">

<platform>Microsoft Windows 98</platform>

<platform>Microsoft Windows 2000</platform>

<platform>Microsoft Windows XP</platform>

<product>CoolWare NET-Suite</product>

</affected>

<description>CoolWare NET-Suite is installed on the

system</description>

</metadata>

*<!--*

*The criteria element specifies the assertion to be tested using*

*information gathered from the endpoint.*

*-->*

<criteria>

*<!--*

*The criterion elements define logical terms in the assertion. This*

*criteria uses 2 criterion elements to check for the presence of CoolWare NET-Suite.*

*By default, the truth values returned by the tests are AND'ed to*

*determine the truth value of the assertion. So this criteria*

*specifies that if both tests return true, then the assertion is*

*true.*

*-->*

<criterion

comment="Any versions of CoolWare iBrowse, prior to version 4, are installed"

test\_ref="oval:tutorial:tst:1"/>

<criterion comment="Specified CoolWare products are installed"

test\_ref="oval:tutorial:tst:2"/>

</criteria>

</definition>

</definitions>

<tests>

*<!--*

*The tests element contains the OVAL Test(s). OVAL Tests specify what to*

*search for on an endpoint (i.e., objects) and what is expected to be*

*found (i.e., states).*

*The registry\_test is used to check information in the*

*Windows registry.*

*-->*

<registry\_test id="oval:tutorial:tst:1" version="4"

comment="Any versions of CoolWare iBrowse, prior to version 4, are installed"

check\_existence="at\_least\_one\_exists" check="all"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_test checks for the correct versions of iBrowse.*

*-->*

<object object\_ref="oval:tutorial:obj:1"/>

<state state\_ref="oval:tutorial:ste:1"/>

</registry\_test>

<registry\_test id="oval:tutorial:tst:2" version="2"

comment="Specified CoolWare products are installed"

check\_existence="at\_least\_one\_exists" check="all"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_test checks that specific CoolWare products are installed.*

*-->*

<object object\_ref="oval:tutorial:obj:2"/>

<state state\_ref="oval:tutorial:ste:2"/>

</registry\_test>

</tests>

<objects>

*<!--*

*The objects element contains the OVAL Object(s).*

*The registry\_object is used to search for information in the Windows*

*registry.*

*-->*

<registry\_object id="oval:tutorial:obj:1" version="3"

comment="The registry key which holds the version of CoolWare iBrowse"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry key containing the installed*

*version of CoolWare iBrowse should be checked.*

*-->*

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SOFTWARE\CoolWare\iBrowse</key>

<name>Version</name>

</registry\_object>

<registry\_object id="oval:tutorial:obj:2" version="1"

comment="The registry key which holds the names of CoolWare products"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry keys that tell which*

*CoolWare products are installed should be checked.*

*-->*

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SOFTWARE\CoolWare</key>

<name>Product</name>

</registry\_object>

</objects>

<states>

*<!--*

*The states element contains the OVAL State(s).*

*The registry\_state is used to describe information expected to be found in*

*the Windows registry.*

*-->*

<registry\_state id="oval:tutorial:ste:1" version="2"

comment="The registry key matches any version of CoolWare iBrowse, prior to version 4"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_state specifies that any values matching the regular expression*

*are expected to be found in the registry.*

*-->*

<value datatype="version" operation="pattern match">^[1-3](\.[0-9])\*$</value>

</registry\_state>

<registry\_state id="oval:tutorial:ste:2" version="3"

comment="The registry key matches CoolWare Products specified below"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_state specifies that the values listed in the referenced*

*variable are expected to be found in the registry.*

*-->*

<value datatype="string" operation="equals" var\_ref="oval:tutorial:var:1"

var\_check=”all”/>

</registry\_state>

</states>

<variables>

*<!--*

*The variables element contains the OVAL Variable(s).*

*-->*

<constant\_variable id="oval:tutorial:var:1" version="1"

datatype="string"

comment="Specific CoolWare products to check for">

*<!--*

*There are different types of variables in OVAL. This is a*

*constant\_variable. A constant\_variable contains a fixed list of values*

*which can be referenced from other OVAL components (e.g., an oval state).*

*The values defined are product names expected to be found in the registry.*

*-->*

<value>iBrowse</value>

<value>eMail</value>

<value>Cool Graphs</value>

<value>Einstein Math Editor</value>

</constant\_variable>

</variables>

</oval\_definitions>

### **Retrieving a File Path from an OVAL Registry Object**

**<?xml version="1.0" encoding="UTF-8"?>**

*<!--*

*- File Object Path Example using a local variable -*

*This example uses a local variable to determine the path used in*

*a file\_object. The local variable is populated from information*

*found in the Windows registry, and is retrieved using a*

*registry\_object.*

*-->*

*<!--*

*The oval\_definitions element is always the root and contains*

*the OVAL Content to be exchanged.*

*Schemas and namespaces needed for OVAL Content relevant to*

*Windows are referenced as attributes in this oval\_defintions element*

*because OVAL components for Windows are required to specify the*

*content.*

*-->*

<oval\_definitions

xsi:schemaLocation="

http://oval.mitre.org/XMLSchema/oval-definitions-5 oval-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#windows windows-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-common-5 oval-common-schema.xsd"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"

xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">

<generator>

*<!--*

*The generator element provides metadata about the tool/application*

*used to develop the OVAL Content.*

*-->*

<oval:product\_name>The OVAL Repository</oval:product\_name>

<oval:schema\_version>5.10.1</oval:schema\_version>

<oval:timestamp>2014-11-17T14:04:44.183-05:00</oval:timestamp>

</generator>

<tests>

*<!--*

*The tests element contains the OVAL Test(s). OVAL Tests specify what to*

*search for on an endpoint (i.e., objects) and what is expected to be*

*found (i.e., states).*

*The file\_test is used to check for the presence of specified files on*

*the endpoint.*

*-->*

<file\_test id="oval:org.mitre.oval:tst:21031" version="4"

comment="mysqld.exe or mysqld-nt.exe exists" check\_existence="at\_least\_one\_exists"

check="all"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

<object object\_ref="oval:org.mitre.oval:obj:11786"/>

</file\_test>

</tests>

<objects>

*<!--*

*The objects element contains the OVAL Object(s).*

*The OVAL registry\_object is used to search for information found in*

*the Windows registry. The OVAL file\_object is used to check for the presence*

*of specified files on the endpoint.*

*-->*

<file\_object id="oval:org.mitre.oval:obj:11786" version="4"

comment="Full path to MySQL executable"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This file\_object specifies locations and file names to check for the*

*presence of the mysql daemon.*

*The potential file locations (path) are defined using a reference to*

*a local variable (oval:org.mitre.oval:var:349).*

*-->*

<path var\_ref="oval:org.mitre.oval:var:349" var\_check="at least one"/>

*<!--*

*The file names (filename) to look for are defined below using a regular*

*expression.*

*-->*

<filename operation="pattern match">^mysqld(|-nt)\.exe$</filename>

</file\_object>

<registry\_object id="oval:org.mitre.oval:obj:11992" version="2"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies the registry keys that tell the*

*locations at which MySQL server is installed and should be checked.*

*-->*

<behaviors windows\_view="32\_bit"/>

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key operation="pattern match">^SOFTWARE\\MySQL AB\\MySQL Server [0-9]\.[0-9]$</key>

<name>Location</name>

</registry\_object>

</objects>

<variables>

*<!--*

*The variables element contains the OVAL Variable(s).*

*-->*

<local\_variable id="oval:org.mitre.oval:var:349" version="3"

comment="Path to MySQL bin directory" datatype="string">

*<!--*

*There are different types of variables in OVAL. This is a*

*local\_variable. A local\_variable references other OVAL components*

*(e.g., an OVAL Object) and derives its values from the referenced*

*component.*

*The values defined in this local variable are derived from a*

*registry\_object (oval:org.mitre.oval:obj:11992). This local\_variable*

*was referenced by file\_object oval:org.mitre.oval:obj:11786 to*

*define the path.*

*-->*

<concat>

<object\_component item\_field="value" object\_ref="oval:org.mitre.oval:obj:11992"/>

<literal\_component>bin</literal\_component>

</concat>

</local\_variable>

</variables>

</oval\_definitions>

### **Patch Example**

**<?xml version="1.0" encoding="UTF-8"?>**

*<!--*

*- Patch Example -*

*This OVAL example checks whether the Microsoft SQL Server*

*2012 Service Pack 1 x64 is installed.*

*The example uses the OVAL registy\_test to check for the correct*

*versions of SQL Server to determine patch status. This example*

*also illustrates the use of extended definitions.*

*-->*

*<!--*

*The oval\_definitions element is always the root and contains*

*the OVAL Content to be exchanged.*

*Schemas and namespaces needed for OVAL components relevant to*

*Windows are referenced as attributes in this oval\_defintions element*

*because these OVAL components are required to specify the OVAL*

*Content.*

*-->*

<oval\_definitions

xsi:schemaLocation="

http://oval.mitre.org/XMLSchema/oval-definitions-5 oval-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#windows windows-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-common-5 oval-common-schema.xsd"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"

xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">

<generator>

*<!--*

*The generator element provides metadata about the tool/application*

*used to develop the OVAL Content.*

*-->*

<oval:product\_name>The OVAL Repository</oval:product\_name>

<oval:schema\_version>5.10.1</oval:schema\_version>

<oval:timestamp>2014-11-05T14:26:16.974-05:00</oval:timestamp>

</generator>

<definitions>

*<!--*

*The definitions element contains the OVAL Defintion(s) to be exchanged.*

*-->*

<definition id="oval:org.mitre.oval:def:20814" version="3" class="patch">

*<!--*

*This definition checks patch status.*

*-->*

<metadata>

*<!--*

*The metadata element contains information about the definition,*

*including its title and description. This definition checks*

*whether the SQL Server service pack is required and whether the*

*service pack has actually has been installed.*

*-->*

<title>MS SQL Server 2012 Service Pack 1 x64 should be installed (KB2674319)</title>

<affected family="windows">

<platform>Microsoft Windows 7</platform>

<platform>Microsoft Windows Server 2003</platform>

<platform>Microsoft Windows Server 2008</platform>

<platform>Microsoft Windows Server 2008 R2</platform>

<platform>Microsoft Windows Vista</platform>

<platform>Microsoft Windows XP</platform>

<product>Microsoft SQL Server 2008</product>

</affected>

*<!--*

*References to resources which provide additional information may also*

*be included in a definition's metadata*

*-->*

<reference source="VENDOR" ref\_id="SQLServer2012SP1-KB2674319-x64-ENU.exe"/>

<reference source="Microsoft" ref\_id="KB2674319"

ref\_url="http://www.microsoft.com/en-us/download/details.aspx?id=35575"/>

<description>MS SQL Server 2012 Service Pack 1 x64 should be installed.</description>

</metadata>

*<!--*

*The criteria specifies the assertion to test using information gathered from the endpoint.*

*-->*

<criteria>

*<!--*

*The criterion and extend\_definition elements define logical terms*

*in the assertion. By default, the truth values returned by the*

*tests are AND'ed to determine the truth value of the assertion. So*

*this criteria specifies that if all criterion/extend\_definition elements*

*return true, then the assertion is true.*

*-->*

<criterion comment="a version of Windows for the x64 architecture is installed"

test\_ref="oval:org.mitre.oval:tst:3653"/>

<extend\_definition comment="Microsoft SQL Server 2012 is installed"

definition\_ref="oval:org.mitre.oval:def:15044"/>

<criterion comment="Check if HKLM\Software\Microsoft\Microsoft SQL Server\$Instance

Names\Setup!Version is less than 11.1.3000.0"

test\_ref="oval:org.mitre.oval:tst:89569"/>

</criteria>

</definition>

<definition id="oval:org.mitre.oval:def:15044" version="12" class="inventory">

*<!--*

*This definition checks software inventory.*

*-->*

<metadata>

*<!--*

*The metadata element contains information about the definition,*

*including its title and description. This definition checks*

*whether the SQL Server service pack has been installed.*

*-->*

<title>Microsoft SQL Server 2012 is installed</title>

<affected family="windows">

<platform>Microsoft Windows 7</platform>

<platform>Microsoft Windows Server 2008</platform>

<platform>Microsoft Windows Server 2008 R2</platform>

<platform>Microsoft Windows Vista</platform>

<product>Microsoft SQL Server 2012</product>

</affected>

*<!--*

*References to resources which provide additional information may also*

*be included in a definition's metadata*

*-->*

<reference source="CPE" ref\_id="cpe:/a:microsoft:sql\_server:2012"/>

<description>Microsoft SQL Server 2012 is installed.</description>

</metadata>

<criteria>

<criterion comment="SQL Server instances with version greater than 11.0.0.0 and less than

12.0.0.0 exist"

test\_ref="oval:org.mitre.oval:tst:79659"/>

</criteria>

</definition>

</definitions>

<tests>

*<!--*

*The tests element contains the OVAL Test(s). OVAL Tests specify what to*

*search for on an endpoint (i.e., objects) and what is expected to be*

*found (i.e., states).*

*The OVAL registry\_test is used to check information found in the*

*Windows registry.*

*-->*

<registry\_test id="oval:org.mitre.oval:tst:3653" version="4"

comment="a version of Windows for the x64 architecture is installed"

check\_existence="at\_least\_one\_exists" check="at least one"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_test checks the system architecture.*

*-->*

<object object\_ref="oval:org.mitre.oval:obj:1576"/>

<state state\_ref="oval:org.mitre.oval:ste:3180"/>

</registry\_test>

<registry\_test id="oval:org.mitre.oval:tst:79659" version="6"

comment="SQL Server instances with version greater than 11.0.0.0 and less than 12.0.0.0

exist"

check\_existence="at\_least\_one\_exists" check="at least one"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_test checks for the correct versions of SQL Server.*

*-->*

<object object\_ref="oval:org.mitre.oval:obj:11792"/>

<state state\_ref="oval:org.mitre.oval:ste:19503"/>

<state state\_ref="oval:org.mitre.oval:ste:19160"/>

</registry\_test>

<registry\_test id="oval:org.mitre.oval:tst:89569" version="1"

comment="Check if HKLM\Software\Microsoft\Microsoft SQL Server\$Instance Names\Setup!Version

is less than 11.1.3000.0"

check\_existence="at\_least\_one\_exists" check="at least one"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_test checks for the correct version of the setup.*

*-->*

<object object\_ref="oval:org.mitre.oval:obj:28893"/>

<state state\_ref="oval:org.mitre.oval:ste:25252"/>

</registry\_test>

</tests>

<objects>

*<!--*

*The objects element contains the OVAL Object(s).*

*The OVAL registry\_object is used to search for information found*

*in the Windows registry.*

*-->*

<registry\_object id="oval:org.mitre.oval:obj:1576" version="1"

comment="This registry key identifies the architecture on the system"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry key containing the*

*endpoint architecture should be checked.*

*-->*

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key>SYSTEM\CurrentControlSet\Control\Session Manager\Environment</key>

<name>PROCESSOR\_ARCHITECTURE</name>

</registry\_object>

<registry\_object id="oval:org.mitre.oval:obj:11792" version="7"

comment="the versions of Microsoft SQL Server instances"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry keys conveyed by*

*both oval:org.mitre.oval:obj:26674 and oval:org.mitre.oval:obj:26798*

*should be checked (both 32 and 64 bit versions)*

*-->*

<set xmlns=http://oval.mitre.org/XMLSchema/oval-definitions-5 set\_operator=”UNION”>

<object\_reference>oval:org.mitre.oval:obj:26798</object\_reference>

<object\_reference>oval:org.mitre.oval:obj:26674</object\_reference>

</set>

</registry\_object>

<registry\_object id="oval:org.mitre.oval:obj:26798" version="5"

comment="the versions of 32 bit Microsoft SQL Server instances"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry key containing the*

*versions of SQL Server (32-bit) should be checked.*

*-->*

<behaviors windows\_view="32\_bit"/>

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key operation="pattern match">^SOFTWARE\\Microsoft\\Microsoft SQL

Server\\MSSQL.\*\..\*\\MSSQLServer\\CurrentVersion$</key>

<name>CurrentVersion</name>

</registry\_object>

<registry\_object id="oval:org.mitre.oval:obj:26674" version="5"

comment="the versions of Microsoft SQL Server instances"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry key containing the*

*versions of SQL Server (64-bit) should be checked. (64 bit architecture*

*is the default.)*

*-->*

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key operation="pattern match">^SOFTWARE\\Microsoft\\Microsoft SQL

Server\\MSSQL.\*\..\*\\MSSQLServer\\CurrentVersion$</key>

<name>CurrentVersion</name>

</registry\_object>

<registry\_object id="oval:org.mitre.oval:obj:28893" version="1"

comment="Registry key HKLM\Software\Microsoft\Microsoft SQL Server\$Instance

Names\Setup!Version exists"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_object specifies that the registry key containing the setup versions*

*for the SQL Server 11 installation should be checked.*

*-->*

<hive>HKEY\_LOCAL\_MACHINE</hive>

<key operation="pattern match">^SOFTWARE\\Microsoft\\Microsoft SQL

Server\\MSSQL11\..\*\\Setup$</key>

<name>Version</name>

</registry\_object>

</objects>

<states>

*<!--*

*The states element contains the OVAL State(s).*

*The OVAL registry\_state is used to describe information expected to be*

*found in the Windows registry.*

*-->*

<registry\_state id="oval:org.mitre.oval:ste:3180" version="4" comment="amd64 architecture"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_state specifies that a value matching the regular expression which*

*determines amd64 architecture is expected to be found in the registry*

*-->*

<value operation="pattern match">^[Aa][Mm][Dd]64$</value>

</registry\_state>

<registry\_state id="oval:org.mitre.oval:ste:19503" version="1"

comment="Version is greater than 11.0.0.0"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_state specifies that a version later than 11 is*

*expected to be found in the registry.*

*-->*

<value datatype="version" operation="greater than">11.0.0.0</value>

</registry\_state>

<registry\_state id="oval:org.mitre.oval:ste:19160" version="1"

comment="Version is less than 12.0.0.0"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_state specifies that a version earlier than 12 is*

*expected to be found in the registry.*

*-->*

<value datatype="version" operation="less than">12.0.0.0</value>

</registry\_state>

<registry\_state id="oval:org.mitre. oval:ste:25252" version="1"

comment="Version is less than 11.1.3000.0"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

*<!--*

*This registry\_state specifies that a version earlier than*

*11.1.3000 is expected to be found in the registry.*

*-->*

<value datatype="version" operation="less than">11.1.3000.0</value>

</registry\_state>

</states>

</oval\_definitions>

#### **Checking for Compliance to a Whitelist**

**<?xml version="1.0" encoding="UTF-8"?>**

*<!--*

*- Linux Whitelist Example -*

*This example uses an OVAL Filter to check the installed software against an*

*approved whitelist of packages. If any software that is not on the*

*whitelist is found, then the endpoint is not in compliance.*

*-->*

*<!--*

*The oval\_definitions element is always the root and contains*

*the OVAL information to be exchanged.*

*Schemas and namespaces needed for OVAL Definitions relevant to*

*Linux are referenced as attributes in this oval\_defintions element*

*because OVAL components for Linux are required to specify the OVAL*

*Content.*

*-->*

<oval\_definitions

xsi:schemaLocation="

http://oval.mitre.org/XMLSchema/oval-definitions-5 oval-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#windows windows-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-common-5 oval-common-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#unix unix-definitions-schema.xsd

http://oval.mitre.org/XMLSchema/oval-definitions-5#linux linux-definitions-schema.xsd"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"

xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5"

xmlns:unix-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix"

xmlns:linux-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux">

*<!--*

*The generator element provides metadata about the tool/application*

*used to develop the OVAL Content.*

*-->*

<generator>

<oval:product\_name>Tutorial Example Generator</oval:product\_name>

<oval:schema\_version>5.11</oval:schema\_version>

<oval:timestamp>2014-12-21T04:42:18.845-05:00</oval:timestamp>

</generator>

<definitions>

*<!--*

*The definitions element contains the OVAL Definition(s) to be exchanged.*

*-->*

<definition id="oval:tutorial:def:1" version="1" class="compliance">

*<!--*

*This definition checks for compliance.*

*-->*

<metadata>

*<!--*

*The metadata element contains information about the definition,*

*including its title and description. This definition checks*

*the installed packages against a whitelist to determine the*

*compliance of the endpoint.*

*-->*

<title>RPM WhiteList</title>

<description>Fail if anything not on the whitelist is installed</description>

</metadata>

<criteria>

*<!--*

*The criteria element specifies the assertion to be*

*tested on information gathered from the endpoint.*

*The criterion elements define logical terms in the assertion.*

*This criterion uses one test, oval:tutorial:tst:1, to check*

*for compliance to the whitelist.*

*-->*

<criterion comment="Test to check that only listed packages are installed."

test\_ref="oval:tutorial:tst:1"/>

</criteria>

</definition>

</definitions>

<tests>

*<!--*

*The tests element contains the OVAL Test(s).*

*The OVAL rpminfo\_test checks the packages installed on a*

*Linux endpoint.*

*-->*

<rpminfo\_test xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux"

id="oval:tutorial:tst:1" version="1" check="all" check\_existence="none\_exist"

comment="all packages">

*<!--*

*This rpminfo\_test specifies that none of the referenced packages*

*(oval:tutorial:obj:2) should be installed (none\_exist) and that*

*all of the packages should be checked (all).*

*-->*

<object object\_ref="oval:tutorial:obj:2"/>

</rpminfo\_test>

</tests>

<objects>

*<!--*

*The objects element contains the OVAL Object(s).*

*The OVAL rpminfo\_object returns a list of installed packages,*

*which match the defined constraints.*

*-->*

<rpminfo\_object xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux"

id="oval:tutorial:obj:2" version="1" comment="Filtered Packages">

*<!--*

*This rpminfo object first finds the names of all installed*

*packages.*

*-->*

<name datatype="string" operation="pattern match">.\*</name>

*<!--*

*Then filters out the package names defined by oval:tutorial:ste:1*

*(i.e., all package names in the whitelist).*

*-->*

<oval-def:filter action="exclude">oval:tutorial:ste:1</oval-def:filter>

</rpminfo\_object>

</objects>

<states>

*<!--*

*The states element contains the OVAL State(s) to be exchanged.*

*The OVAL rpminfo\_state is to specify the packages expected to be*

*found on the endpoint.*

*-->*

<rpminfo\_state id="oval:tutorial:ste:1" version="1"

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux">

*<!--*

*This rmpinfo\_state defines a list of values equivalent to those*

*specified in the referenced OVAL Variable, oval:tutorial:var:1.*

*-->*

<name datatype="string" operation="equals" var\_ref="oval:tutorial:var:1"/>

</rpminfo\_state>

</states>

<variables>

*<!--*

*The variables element contains the OVAL Variable(s) to be exchanged.*

*-->*

<constant\_variable id="oval:tutorial:var:1" version="1" datatype="string"

comment="Package Names">

*<!--*

*There are different types of variables in OVAL. This is a*

*constant\_variable. A constant\_variable contains a fixed list of values*

*which can be referenced from other OVAL components (e.g., an oval state).*

*This OVAL Variable defines the whitelist as the list of approved*

*packages which are permitted to be installed on the endpoint.*

*These package names in the whitelist are defined by the variable's*

*value elements below.*

*-->*

<value>termcap</value>

<value>auditd</value>

<value>libselinux</value>

*<!--*

*More value elements may be added to further define the approved whitelist*

*of packages which may be installed on the endpoint.*

*-->*

</constant\_variable>

</variables>

</oval\_definitions>

## **Additional Resources**

* OVAL Author’s Resources http://ovalproject.github.io/documentation/repository/authorsresources This page gathers documents and tools for authoring content in the OVAL Language into a single location. Included are prerequisites, instructional documents, useful tools, and how to obtain further assistance.
* OVAL Language Revision Process http://ovalproject.github.io/documentation/policy/revisionprocess/ This page details how the OVAL Language changes and evolves, including the four major milestones for creating a new version of the OVAL Language.
* Requesting Changes to the OVAL Language http://ovalproject.github.io/documentation/policy/changerequests/ This page gives guidelines to help OVAL Community members propose changes to the OVAL Language, including requests to add new OVAL Constructs (e.g., component schemas, core capabilities, tests, entities, or functions), improve existing OVAL Constructs, and/or deprecate OVAL Constructs.
* Validating an OVAL Document http://ovalproject.github.io/documentation/validation/ This page explains how to validate an OVAL document to ensure a common and expected structure amongst OVAL documents being passed between different users.
* OVAL Utilities http://ovalutils.sourceforge.net/ Project for developing a set of utilities for manipulating content written in the OVAL Language. These are general utilities that will assist anyone in using OVAL content.
* OVAL Repository Style Guide http://ovalproject.github.io/documentation/repository/styleguide/ This page details style guidelines and best practices for writing OVAL content.
* Submitting Content http://ovalproject.github.io/documentation/repository/ This page provides guidelines for submitting new and modified content to the OVAL Repository (http://oval.mitre.org/repository/).

Copyright 2017 Center for Internet Security. [Terms of Use](http://oval.mitre.org/about/termsofuse.html)