CIS Benchmarks

Center for Internet Security Configuration Assessment Tool CIS-CAT

OVAL Customization Guide

v3.0.43

October 24, 2017

Table of Contents

Introduction

The purpose of this document is to describe the components of the CIS-CAT XML Benchmarks which utilize the OVAL checking language, and provide instruction on how to perform common customizations to a given CIS-CAT XML Benchmark. While not required, it is recommended that you use a XML editor when working with CIS-CAT XML Benchmarks. All CIS-CAT XML Benchmark are located in the `benchmarks` directory within the CIS-CAT ZIP archive. If you need support customizing CIS-CAT's policies, or have suggestions for how this guide can be improved, please feel free to post a question/comment to the CIS-CAT Discussion forum or send an email to support@cisecurity.org. For a primer in XML basics please review the following tutorials:

- http://www.w3schools.com/xml/xml_tree.asp
- http://www.w3schools.com/xml/xml_syntax.asp
- http://www.w3schools.com/xml/xml_elements.asp
- http://www.w3schools.com/xml/xml_attributes.asp
- http://www.w3schools.com/xml/xml_dtd.asp
- http://www.w3schools.com/xml/xml_validator.asp

The intent of this guide is to introduce the structure of the CIS OVAL-based XML benchmarks, as well as provide sets of instructions to perform various customizations of those benchmarks.

OVAL Implementation

The Open Vulnerability and Assessment Language (OVAL) is used to identify vulnerabilities and issues. Common examples of the use of OVAL files are:

- the checking language referenced from a separate XCCDF file,
- the checking language referenced from a checklist component of a SCAP source data stream,
- the checking language referenced from a CPE dictionary component of SCAP source data stream

The OVAL component will contain the definitions, tests, as well as the state a target system is expected to exhibit. When CIS-CAT encounters a reference to an OVAL definition, it parses the specific OVAL components/files and uses those referenced definition identifiers to look up the appropriate tests to be executed. Each OVAL definition may be comprised of one-to-many OVAL tests; the results of which may be logically combined to enumerate an overall definition result. The CIS-CAT evaluation engine is the controller for parsing the required tests, collecting the appropriate system characteristics, evaluating the collected information against the expected state, and recording the success, failure, or any error conditions of a given test. CIS-CAT supports components specified using versions 5.3, 5.8, 5.10.1, and 5.11 of the OVAL language.

The full list of CIS-CAT supported OVAL tests can be found in the CIS-CAT User's Guide.

Benchmark Composition - OVAL

CIS benchmark content which utilizes the OVAL checking language is currently distributed as a set of 4 component files:

1. XCCDF – The XCCDF component of the benchmark will be structured in the same manner as is detailed in the CIS XML Policy Customization Guide. The XCCDF contains the various profiles which may be assessed using CIS-CAT, as well as each benchmark recommendation – known as a <Rule>. These <Rule> elements will contain references to definitions located in the OVAL component.
2. OVAL – The OVAL component of the benchmark contains the set of definitions referenced by the XCCDF component. These definitions are structured such that any number of criteria may be evaluated in order to produce an overall result.
3. CPE-Dictionary – The CPE-Dictionary component contains information regarding the relevant platforms to which the benchmark is applicable. An entire benchmark, a group of recommendations, or a single recommendation may only be applicable to a particular operating system or application (such as Windows 7 or Internet Explorer 10). The CPE-Dictionary defines these platforms, and will contain references to definitions located in the CPE-OVAL component.
4. CPE-OVAL – The CPE-OVAL component of the benchmark is structured in the same manner as the OVAL component; containing the definitions necessary to evaluate the various CPE items contained in the CPE-Dictionary.

At a minimum, when using an OVAL-based benchmark, the XCCDF and OVAL components are required. If a set of recommendations does not apply to any specific platform or operating system, the CPE-Dictionary and CPE-OVAL components would not be necessary.

The Profile, Group, and Value compositions in OVAL-based benchmarks are the same as in ECL-based benchmarks. See the CIS-CAT Policy Customization Guide for information regarding the construction of customized Profiles, Groups, and/or Values.

Rule Composition - OVAL

An `xccdf:Rule` is a basic building block of all CIS-CAT XML Benchmarks. Each `xccdf:Rule` corresponds with one or more recommendations made in a CIS Benchmark. Below is an example of an `xccdf:Rule`:

```
 1  <xccdf:Rule id="sample_xccdf_rule_1">
 2     <xccdf:title>Sample XCCDF Rule #1</xccdf:title>
 3     <xccdf:description>
 4         This Rule exemplifies a basic XCCDF Rule
 5     </xccdf:description>
 6     <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
 7        <xccdf:check-export export-name=”…” value-id=”…”/>
 8        <xccdf:check-content-ref href=”…” name=”…”/>
 9     </xccdf:check>
10  </xccdf:Rule>
11
```

Line 1

Opening tag for xccdf:Rule. Every xccdf:Rule must have an id attribute that is set to a unique value. In the above example, the id attribute is set to sample_xccdf_rule_1. No other xccdf:Rule element may have this id. xccdf:Profiles are defined by referencing these id attributes.

Line 2

The title of the xccdf:Rule. The title is typically used to refer to a given xccdf:Rule in reports or other documentation. Titles are typically written in a manner that conveys the intent of the Rule, such as:

- Ensure /etc/password is owned by root:root
- Ensure the Banner directive is set in sshd_config

Lines 3-5

The description of the xccdf:Rule. The xccdf:description typically provides information about the configuration subject, such as a registry key or network service. It also provides other information such as security benefit statements, procedures for implementing/auditing the recommended state, and potential impacts associated with implementing the recommended configuration state.

Line 6

The xccdf:check element for an OVAL-based <Rule> will reference a supported OVAL check system. CIS-CAT currently only supports the http://oval.mitre.org/XMLSchema/oval-definitions-5 check system.

Line 7

The xccdf:check-export element allows for the utilization of specific values when assessing the associated OVAL definitions. These values are referenced by the value-id attribute, and can be tailored using the xccdf:refine-value element within each profile. The export-name attribute defines the OVAL variable ID used when evaluating the associated OVAL definition.

Line 8

The xccdf:check-content-ref element specifies the file location and unique identifier of the OVAL definition(s) to be evaluated in order to determine the overall Pass/Fail result status for this xccdf:Rule. The href attribute specifies the name of the file in which applicable OVAL definition(s) are found. The optional name attribute indicates the unique ID for an OVAL definition to be evaluated. If no name is specified, ALL OVAL definitions found in the file denoted by the href will be evaluated.

Line 10

Closing tag for xccdf:Rule.

OVAL Structure

When a CIS benchmark XCCDF utilizes the OVAL check system, the benchmark's `<Rule>` elements will contain a `<check-content-ref>` element, as described in the previous section. The `<check-content-ref>` element's `href` attribute will specify the path to and name of an OVAL definitions file, containing all constructs necessary to evaluate the `<Rule>`.

The full OVAL schema can be found at http://oval.mitre.org/language/version5.10.1/. The "core" and "common" schemas describe the base constructs implemented in order to support evaluation using the OVAL check system. As noted above, CIS-CAT supports various test types in the following schema extensions:

- Independent
- Microsoft Windows
- Linux
- UNIX

Overall OVAL Definitions File Structure

In general, an OVAL definitions file is constructed of 6 sections, explained below.

```
<oval_definitions ...>

   <generator>
      The generator element contains information regarding the product name,
      schema version, and date/time the OVAL definitions were generated.
   </generator>

   <definitions>
      This section contains individual <definition> elements and the various
      <criteria> used in their evaluation.
   </definitions>

   <tests>
      This section contains the specific tests used as criterion.

      Examples include:
         <family_test>
         <registry_test>
         <file_test>
         <passwordpolicy_test>
         And many more – see the supported test types above
   </tests>

   <objects>
      This section contains the specific objects used in the collection of system
      characteristics used to compare against expected states or in existence checks.

      Examples include:
         <family_object>
         <registry_object>
         <file_object>
         <passwordpolicy_object>
         And many more – see the supported test types above
   </objects>

   <states>
      This section contains the expected states required when assessing target systems.

      Examples include:
         <family_state>
         <registry_state>
         <file_state>
         <passwordpolicy_state>
         And many more – see the supported test types above
   </states>

   <variables>
      This section contains any local, constant, or external variables used in the
      evaluation of any test, object, or expected state.

      Examples include:
         <local_variable>
         <constant_variable>
         <external_variable>
   </variables>
</oval_definitions>
```

Definitions

An OVAL definition is the entry point by which an XCCDF `<Rule>` is evaluated using OVAL. The definition lays out the various criteria to be evaluated in order to determine if the system under test matches some expected state. A definition can reference other definitions, and may logically combine the results from any number of criteria culminating in an overall result for the definition. Criteria referenced by a definition are in the form of OVAL Tests.

```
<definition
   class="compliance"
   id="oval:org.cisecurity.oval:def:1002"
   version="1">
    <metadata>
       <title>Set 'Configure minimum PIN length for startup' to 'Enabled:7'</title>
       <description>
           Set 'Configure minimum PIN length for startup' to 'Enabled:7'
       </description>
    </metadata>

    <criteria operator=”AND”>
        <criterion
           comment="Set ‘Configure minimum PIN length for startup’ to ‘Enabled:7’"
          test_ref="oval:org.cisecurity.oval:tst:1002"/>
    </criteria>
</definition>
```

The unique identifier for the definition must use the following format:
`oval:[reverse_fqdn].oval:def:[unique_numeric_value]`

The “reverse_fqdn” refers to the fully qualified domain name of an organization, in reverse. In the template above, the “reverse_fqdn” of `org.cisecurity` represents the FQDN for CIS, `cisecurity.org`.

A root `<criteria>` element must always be present, and may contain nested criteria or references to an individual test, called a `<criterion>`. Multiple `<criterion>` elements can be contained within a single `<criteria>`. A result for the `<criteria>` is calculated by logically combining the results from each `<criterion>` according to the operator attribute of the `<criteria>`. The acceptable values for the operator are:

- AND – The AND operator produces a true result if all <criterion> or sub-<criteria> element results are true

- ONE – The ONE operator produces a true result if one and only one <criterion> or sub-<criteria> element result is true
- OR – The OR operator produces a true result if one or more <criterion> or sub-<criteria> element results are true
- XOR – The XOR operator produces a true result if an odd number of <criterion> or sub-<criteria> element results are true

A `<criterion>` contains the `test_ref` attribute, essentially linking this `<definition>` to its associated `test`(s).

Tests

An OVAL Test is used to compare an object against any number of expected states (including zero expected states, known as an "existence-only test").

```
<registry_test
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   check="all"
   check_existence="at_least_one_exists"
   comment="..."
   id="oval:org.cisecurity.oval:tst:1002"
   version="1">
    <object object_ref="oval:org.cisecurity.benchmarks.o_microsoft:obj:1002"/>
    <state state_ref="oval:org.cisecurity.oval:ste:1002"/>
</registry_test>
```

The unique identifier for the test must use the following format:
`oval:[reverse_fqdn].oval:tst:[unique_numeric_value]`

The "reverse_fqdn" refers to the fully qualified domain name of an organization, in reverse. In the template above, the "reverse_fqdn" of `org.cisecurity` represents the FQDN for CIS, `cisecurity.org`.

Objects

An OVAL Object defines the specific information to be collected from the system under test. The items collected which match the object information are known as system characteristics, and are subsequently used by an OVAL Test when comparing against an expected state.

```
<registry_object
   xmlns=http://oval.mitre.org/XMLSchema/oval-definitions-5#windows
   comment="..."
   id="oval:org.cisecurity.oval:obj:1002"
   version="1">

    <hive>HKEY_LOCAL_MACHINE</hive>
    <key>Software\Policies\Microsoft\FVE</key>
    <name>MinimumPIN</name>
</registry_object>
```

The unique identifier for the object must use the following format:
`oval:[reverse_fqdn].oval:obj:[unique_numeric_value]`

The "reverse_fqdn" refers to the fully qualified domain name of an organization, in reverse. In the template above, the "reverse_fqdn" of `org.cisecurity` represents the FQDN for CIS, `cisecurity.org`.

States

An OVAL State defines the expected characteristics of the system under test.

```
<registry_state
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   comment="..."
   id="oval:org.cisecurity.oval:ste:1002"
   version="1">

    <type>reg_dword</type>
    <value
       datatype="int"
       operation="equals"
       var_ref="oval:org.cisecurity.oval:var:1002"/>
</registry_state>
```

The unique identifier for the state must use the following format:
`oval:[reverse_fqdn].oval:ste:[unique_numeric_value]`

The "reverse_fqdn" refers to the fully qualified domain name of an organization, in reverse. In the template above, the "reverse_fqdn" of `org.cisecurity` represents the FQDN for CIS, `cisecurity.org`.

Variables

An OVAL Variable may be used to define three types of information:

1. A piece of information which may be specific to the system under test, such as references to installed file locations,
2. A piece of information which may be provided from an external source, such as the XCCDF benchmark being assessed,
3. A defined constant value

The unique identifier for the variable must use the following format:
`oval:[reverse_fqdn].oval:var:[unique_numeric_value]`

The "reverse_fqdn" refers to the fully qualified domain name of an organization, in reverse. In the template above, the "reverse_fqdn" of `org.cisecurity` represents the FQDN for CIS, `cisecurity.org`.

OVAL Recipes

The following OVAL recipes are designed to provide some simple examples of OVAL content which may be added or customized for an organization's needs. The recipes provided are not intended to be a comprehensive list of capabilities provided by OVAL. The OVAL website should be considered the authoritative source for and most comprehensive information available regarding the myriad capabilities of OVAL. Contact CIS member support at support@cisecurity.org with specific questions regarding the customization of OVAL benchmarks, or to report issues.

General Recipes

Removing the Platform verification

Each OVAL-based CIS benchmark includes a verification of the platform applicable to the benchmark. For example, the CIS Microsoft Windows 8 benchmark contains a platform applicability check ensuring the benchmark recommendations are being assessed against a Microsoft Windows 8 platform. If a user attempts to assess this benchmark against a different platform, that user will be prompted with a message indicating the platform mismatch, and all recommendations in that benchmark will be evaluated to "Not Applicable".

A user can disable this platform check with a simple edit of the XML in the benchmark XCCDF file. The `<platform>` element is typically located near the beginning of the benchmark XCCDF, following the `<notice>` element (containing the terms of use), and prior to the `<version>` element indicating the major and minor version number of the benchmark.

```
<notice id="terms-of-use" xml:lang="en">
    BACKGROUND. The Center for Internet Security ("CIS") provides benchmarks...
</notice>
<platform idref="cpe:/o:microsoft:windows_7"/>
<version>2.1.0.2</version>
```

To disable the platform verification, simply comment out the highlighted line:

```
<notice id="terms-of-use" xml:lang="en">
    BACKGROUND. The Center for Internet Security ("CIS") provides benchmarks...
</notice>
<!-- <platform idref="cpe:/o:microsoft:windows_7"/> -->
<version>2.1.0.2</version>
```

The next time CIS-CAT is executed, the benchmark's platform verification will be skipped.

Modifying the definition being evaluated by a Rule

If a custom `<definition>` has been created, then a `<Rule>` will need to be created or modified in order for that `<definition>` to be assessed as part of a CIS benchmark. Using an existing `<Rule>` as a template, the `href` and `name` attributes of the `<check-content-ref>` element are those required to link to the new `<definition>`.

Using the following <Rule> as an example:

```
<Rule
   id="xccdf_org.cisecurity.benchmarks_rule_Account_lockout_threshold"
   selected="false"
   role="full"
   weight="1.0">

  <title xml:lang="en">
    Set 'Account lockout threshold' to '5 invalid logon attempt(s)'
  </title>
  <description xml:lang="en">This policy setting [...]</description>
  ...
  <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-export
      export-name=
         "oval:org.cisecurity.benchmarks.microsoft_windows_server_2012:var:1002"
      value-id="xccdf_org.cisecurity.benchmarks_value_1.1.1.1.2_var"/>
    <check-content-ref
       href="CIS_Microsoft_Windows_Server_2012_Benchmark_v1.0.0-oval.xml"
       name="oval:org.cisecurity.benchmarks.microsoft_windows_server_2012:def:1002"/>
  </check>
</Rule>
```

Ensure the href attribute denotes the OVAL definitions file in which the custom <definition> has been created.
Ensure the name attribute denotes the appropriate unique id of the custom <definition>.

Modifying a Recommended Value (XCCDF and OVAL)

Organizations may have specific operational needs to modify the value against which the benchmark is checking. For example, an organizational requirement may be to have an account be locked out after only 4 invalid login attempts, instead of the benchmark-recommended value of 5. In this scenario, the OVAL definitions don't necessarily need to be modified, only the value being assessed against. The following steps describe how to find the value the benchmark is using, and how to make the necessary modifications.

Find the Recommendation in the XML file, using the title in the benchmark. For example, search for Set 'Account lockout threshold' to '5 invalid logon attempt(s)'.

In this case, you will find a <Rule> which looks like this:

```
<Rule
   id="xccdf_org.cisecurity.benchmarks_rule_Account_lockout_threshold"
   selected="false"
   role="full"
   weight="1.0">

  <title xml:lang="en">
    Set 'Account lockout threshold' to '5 invalid logon attempt(s)'
  </title>
  <description xml:lang="en">This policy setting [...]</description>
  ...
  <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-export
      export-name=
                "oval:org.cisecurity.benchmarks.microsoft_windows_server_2012:var:1002"
      value-id="xccdf_org.cisecurity.benchmarks_value_1.1.1.1.2_var"/>
    <check-content-ref
       href="CIS_Microsoft_Windows_Server_2012_Benchmark_v1.0.0-oval.xml"
       name="oval:org.cisecurity.benchmarks.microsoft_windows_server_2012:def:1002"/>
  </check>
</Rule>
```

Edit the `<Rule>`'s `<title>` and/or `<description>` so CIS-CAT output and results will reflect the customized value.

Note the "`value-id`" attribute – this is a reference to the value being utilized by the benchmark. Next, search the benchmark for that "`value-id`"; An `<xccdf:Value>` with that "`id`" will be found:

```
<Value id="xccdf_org.cisecurity.benchmarks_value_1.1.1.1.2_var"
     type="number"
 operator="equals">

  <title>Set 'Account lockout threshold' to '5'</title>
  <description>
    This variable is used in Benchmark item "Set 'Account lockout threshold' to
   '5 invalid logon attempt(s)'" -&gt; "Set 'Account lockout threshold' to '5'
  </description>

  <value>5</value>
</Value>
```

Modify the `<value>` to the desired value. For example, if the desired value is 4, the updated `<value>` will appear as:

```
<value>4</value>
```

Edit the `<Value>`'s `<title>` and/or `<description>` so CIS-CAT output and results will reflect the customized value.

Platform-Independent Recipes

The following recipes/templates are available for use against any target platform which CIS supports. Use of platform independent tests requires tests, objects, and states utilizing the platform independent namespace, `xmlns=http://oval.mitre.org/XMLSchema/oval-definitions-5#independent`. The full list of platform-independent tests, their descriptions, required test, object, and state information can be found here.

Assessing the Operating System Family

In the OVAL definitions file, inside the `<definitions>` section, create a `<definition>`.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Microsoft Windows 7 is installed</title>
        <affected family="windows">
            <platform>Microsoft Windows 7</platform>
        </affected>
        <description>
            The operating system installed on the system is Microsoft Windows 7.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="the operating system is part of the Microsoft Windows family"
          test_ref="oval:org.cisecurity.oval:tst:99"/>
      </criteria>
    </definition>
```

Note the value for the "`test_ref`" attribute. This needs to be a unique identifier for the `<family_test>`.

Inside the `<tests>` section, create a `<family_test>` using the platform independent namespace.

```
<family_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="the installed operating system is part of the Microsoft Windows family"
   check_existence="at_least_one_exists"
   check="only one"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
     <state state_ref="oval:org.cisecurity.oval:ste:99"/>

</family_test>
```

Note the values for the "`object_ref`" and "`state_ref`" attributes. These need to be unique identifiers for the `<family_object>` and `<family_state>`, respectively.

Inside the `<objects>` section, create a `<family_object>` using the platform independent namespace.

```
<family_object
```

```
   id="oval:org.cisecurity.oval:obj:99" version="1"
   comment="This is the default family object. Only one family object should exist."
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"/>
```

NOTE: The `<family_object>` is an “empty object”, meaning there are no child elements.

Inside the `<states>` section, create a `<family_state>` using the platform independent namespace.

```
<family_state
   id="oval:org.cisecurity.oval:ste:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"
   version="2"
   comment="Microsoft Windows family">

      <family operation="case insensitive equals">windows</family>

</family_state>
```

The allowed values for the `<family>` element are:

- catos
- ios
- macos
- pixos
- undefined
- unix
- vmware_infrastructure
- windows

Assessing an Environment Variable Value

In the OVAL definitions file, inside the `<definitions>` section, create a `<definition>`.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Check an Environment Variable</title>
        <description>
            Check an Environment Variable.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="Check an Environment Variable"
          test_ref="oval:org.cisecurity.oval:tst:99"/>
      </criteria>
    </definition>
```

Note the value for the “`test_ref`” attribute. This needs to be a unique identifier for the `<environmentvariable_test>`.

Inside the `<tests>` section, create an `<environmentvariable_test>` using the platform independent namespace.

```
<environmentvariable_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="check and environment variable"
   check_existence="at_least_one_exists"
   check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
     <state state_ref="oval:org.cisecurity.oval:ste:99"/>

</environmentvariable_test>
```

Note the values for the "`object_ref`" and "`state_ref`" attributes. These need to be unique identifiers for the `<environmentvariable_object>` and `<environmentvariable_state>`, respectively.

Inside the `<objects>` section, create an `<environmentvariable_object>` using the platform independent namespace.

```
<environmentvariable_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"
   version="1">

     <name operation="equals">JAVA_HOME</name>
</environmentvariable_object>
```

Note that the `<name>` element is mandatory and specifies which environment variable to collect from the target system.

Inside the `<states>` section, create an `<environmentvariable_state>` using the platform independent namespace.

```
<environmentvariable_state
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#independent"
   id="oval:org.cisecurity.oval:ste:99"
   version="1">

    <name datatype="string" operation="equals">JAVA_HOME</name>
    <value datatype="string" operation="equals">C:\Java</value>
</environmentvariable_state>
```

In order to assess the `<environmentvariable_state>`, at least one (or both) of the child elements must be included. The `<name>` element describes the name of the environment variable, and the `<value>` element describes the actual value of the specified environment variable.

Microsoft Windows Recipes

The following recipes/templates are available for use against Microsoft Windows platforms. Use of Microsoft Windows platform tests requires tests, objects, and states utilizing the Microsoft Windows namespace, xmlns=http://oval.mitre.org/XMLSchema/oval-definitions-5# windows. The full list of Microsoft Windows tests, their descriptions, required test, object, and state information can be found here.

Test if a File Exists

Testing for the existence of a file does not require the creation of an expected state. Attributes defined on the Windows <file_test> construct allow for the testing of a file's existence.

In the OVAL definitions file, inside the <definitions> section, create a <definition>.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Ensure a file exists</title>
        <description>
            Ensure a file exists.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="Ensure a file exists"
          test_ref="oval:org.cisecurity.oval:tst:99"/>
      </criteria>
    </definition>
```

Note the value for the "test_ref" attribute. This needs to be a unique identifier for the <file_test>.

Inside the <tests> section, create a <file_test> using the Microsoft Windows namespace.

```
<file_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="ensure a file exists"
   check_existence="at_least_one_exists"
   check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
</file_test>
```

The check_existence attribute of the <file_test> is the key for testing for the existence of a file. Because no expected state is referenced in the test (there is no <state state_ref="…"/> element), CIS-CAT will collect the items based on the object information, and the test result will be based on evaluating the number of collected items against the value of the check_existence attribute. The following are acceptable values for the check_existence attribute:

- all_exist – Every item being collected exists on the target system

- any_exist – Any number of items (zero or more) collected exist on the target system
- at_least_one_exists – At least one item (1 or more) must exist on the target system
- none_exist – Zero items may exist on the target system
- only_one_exists – One and only one item must exist on the target system

Note the value for the "object_ref" attribute. This needs to be unique identifier for the <file_object> being collected from the target system.

Inside the <objects> section, create a <file_object> using the Microsoft Windows namespace.

```
<file_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   version="1">

    <filepath>C:\Program Files\Internet Explorer\iexplore.exe</filepath>
</file_object>

OR

<file_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   version="1">

     <path>C:\Program Files\Internet Explorer</path>
     <filename>iexplore.exe</filename>

</file_object>
```

There are two methods by which a <file_object> can specify the file(s) to be collected:

1. Using the <filepath> describes the full path to the file.
2. Using the <path> and <filename> allows for more generalized listing of paths, regular expression pattern matching, etc., when searching for files.

Test if a Registry Key Exists

Testing for the existence of a registry key does not require the creation of an expected state. Attributes defined on the Windows <registry_test> construct allow for the testing of a registry key's existence.

In the OVAL definitions file, inside the <definitions> section, create a <definition>.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Ensure a registry key exists</title>
        <description>
            Ensure a registry key exists.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="Ensure a registry key exists"
```

```
          test_ref="oval:org.cisecurity.oval:tst:99"/>
    </criteria>
</definition>
```

Note the value for the "test_ref" attribute. This needs to be a unique identifier for the <registry_test>.

Inside the <tests> section, create a <registry_test> using the Microsoft Windows namespace.

```
<registry_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="ensure a registry key exists"
   check_existence="at_least_one_exists"
   check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
</file_test>
```

The check_existence attribute of the <registry_test> is the key for testing for the existence of a registry key. Because no expected state is referenced in the test (there is no <state state_ref="…"/> element), CIS-CAT will collect the items based on the object information, and the test result will be based on evaluating the number of collected items against the value of the check_existence attribute. The following are acceptable values for the check_existence attribute:

- all_exist – Every item being collected exists on the target system
- any_exist – Any number of items (zero or more) collected exist on the target system
- at_least_one_exists – At least one item (1 or more) must exist on the target system
- none_exist – Zero items may exist on the target system
- only_one_exists – One and only one item must exist on the target system

Note the value for the "object_ref" attribute. This needs to be unique identifier for the <registry_object> being collected from the target system.

Inside the <objects> section, create a <registry_object> using the Microsoft Windows namespace.

```
<registry_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   version="1">

    <hive>HKEY_LOCAL_MACHINE</hive>
    <key>Software\Policies\Microsoft\FVE</key>
    <name>MinimumPIN</name>
</file_object>
```

The three child elements of a <registry_object> are mandatory:

<hive> indicates the registry hive the key belongs to. This is restricted to a specific set of values:

- HKEY_CLASSES_ROOT,
- HKEY_CURRENT_CONFIG,
- HKEY_CURRENT_USER,
- HKEY_LOCAL_MACHINE, and
- HKEY_USERS

`<key>` describes a registry key within the <hive> to be collected.
`<name>` describes the name assigned to a value associated with a registry key.

Test if a Registry Value is greater than an Expected Value

In the OVAL definitions file, inside the `<definitions>` section, create a `<definition>`.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Ensure a registry key value is greater than 7</title>
        <description>
            Ensure a registry key value is greater than 7.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="Ensure a registry key value is greater than 7"
          test_ref="oval:org.cisecurity.oval:tst:99"/>
    </criteria>
</definition>
```

Note the value for the "`test_ref`" attribute. This needs to be a unique identifier for the `<registry_test>`.

Inside the `<tests>` section, create a `<registry_test>` using the Microsoft Windows namespace.

```
<registry_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="Ensure a registry key value is greater than 7"
   check_existence="at_least_one_exists"
   check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
     <state state_ref=”oval:org.cisecurity.oval:ste:99”/>
</file_test>
```

Inside the `<objects>` section, create a `<registry_object>` using the Microsoft Windows namespace.

```
<registry_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   version="1">

    <hive>HKEY_LOCAL_MACHINE</hive>
    <key>Software\Policies\Microsoft\FVE</key>
    <name>MinimumPIN</name>
</registry_object>
```

The three child elements of a `<registry_object>` are mandatory:

`<hive>` indicates the registry hive the key belongs to. This is restricted to a specific set of values:

- HKEY_CLASSES_ROOT,
- HKEY_CURRENT_CONFIG,

- HKEY_CURRENT_USER,
- HKEY_LOCAL_MACHINE, and
- HKEY_USERS

<key> describes a registry key within the <hive> to be collected.

<name> describes the name assigned to a value associated with a registry key.

Inside the <states> section, create an <registry_state> using the Microsoft Windows namespace.

```
<registry_state
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
   id="oval:org.cisecurity.oval:ste:99"
   version="1">

    <type>reg_dword</type>
    <value datatype="int" operation="greater than">7</value>
</registry_state>
```

The <registry_state> describes the expected values of various characteristics of the collected items. In this example, the collected registry item must have a type of reg_dword, and a value greater than 7 in order to yield a passing test result.

Unix Recipes

The following recipes/templates are available for use against any "Unix flavored" platforms, such as Red Hat Linux, AIX, CentOS, Debian, etc. Use of generic Unix platform tests requires tests, objects, and states utilizing the generic Unix namespace, `xmlns=http://oval.mitre.org/XMLSchema/oval-definitions-5# unix.` The full list of Unix tests, their descriptions, required test, object, and state information can be found here.

Test if a File Exists

Testing for the existence of a file does not require the creation of an expected state. Attributes defined on the Unix `<file_test>` construct allow for the testing of a file's existence.

In the OVAL definitions file, inside the `<definitions>` section, create a `<definition>`.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Ensure a file exists</title>
        <description>
            Ensure a file exists.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="Ensure a file exists"
          test_ref="oval:org.cisecurity.oval:tst:99"/>
      </criteria>
    </definition>
```

Note the value for the "`test_ref`" attribute. This needs to be a unique identifier for the `<file_test>`.

Inside the `<tests>` section, create a `<file_test>` using the Unix namespace.

```
<file_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="ensure a file exists"
   check_existence="at_least_one_exists"
   check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
</file_test>
```

The `check_existence` attribute of the `<file_test>` is the key for testing for the existence of a file. Because no expected state is referenced in the test (there is no `<state state_ref="…"/>` element), CIS-CAT will collect the items based on the `object` information, and the test result will be based on evaluating the number of collected items against the value of the `check_existence` attribute. The following are acceptable values for the `check_existence` attribute:

- `all_exist` – Every item being collected exists on the target system

- `any_exist` – Any number of items (zero or more) collected exist on the target system
- `at_least_one_exists` – At least one item (1 or more) must exist on the target system
- `none_exist` – Zero items may exist on the target system
- `only_one_exists` – One and only one item must exist on the target system

Note the value for the "`object_ref`" attribute. This needs to be unique identifier for the `<file_object>` being collected from the target system.

Inside the `<objects>` section, create a `<file_object>` using the Unix namespace.

```
<file_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix"
   version="1">

    <filepath>/var/home/test/scratchpad.txt</filepath>
</file_object>

------ OR ------

<file_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#unix"
   version="1">

     <path>/var/home/test</path>
     <filename>scratchpad.txt</filename>

</file_object>
```

There are two methods by which a <file_object> can specify the file(s) to be collected:

3. Using the `<filepath>` describes the full path to the file.
4. Using the `<path>` and `<filename>` allows for more generalized listing of paths, regular expression pattern matching, etc., when searching for files.

Linux Recipes

The following recipes/templates are available for use against Linux platforms, such as Red Hat Linux. Use of Linux platform tests requires tests, objects, and states utilizing the Linux namespace, `xmlns=http://oval.mitre.org/XMLSchema/oval-definitions-5# linux.` The full list of Linux tests, their descriptions, required test, object, and state information can be found here.

Test if an RPM Package Exists

Testing for the existence of a Linux RPM does not require the creation of an expected state. Attributes defined on the Linux `<rpminfo_test>` construct allow for the testing of a RPM's existence.

In the OVAL definitions file, inside the `<definitions>` section, create a `<definition>`.

```
<definition
   id="oval:org.cisecurity.oval:def:99"
   version="1"
   class="compliance">

    <metadata>
        <title>Ensure a RPM exists</title>
        <description>
            Ensure a RPM exists.
        </description>
    </metadata>

    <criteria>
        <criterion
           comment="Ensure a RPM exists"
          test_ref="oval:org.cisecurity.oval:tst:99"/>
      </criteria>
    </definition>
```

Note the value for the "`test_ref`" attribute. This needs to be a unique identifier for the `<rpminfo_test>`.

Inside the `<tests>` section, create a `<rpminfo_test>` using the Linux namespace.

```
<rpminfo_test
   id="oval:org.cisecurity.oval:tst:99"
   version="1"
   comment="ensure a file exists"
   check_existence="at_least_one_exists"
   check="all"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux">

     <object object_ref="oval:org.cisecurity.oval:obj:99"/>
</rpminfo_test>
```

The `check_existence` attribute of the `<rpminfo_test>` is the key for testing for the existence of a RPM. Because no expected state is referenced in the test (there is no `<state state_ref="…"/>` element), CIS-CAT will collect the items based on the `object` information, and the test result will be based on evaluating the number of collected items against the value of the `check_existence` attribute. The following are acceptable values for the `check_existence` attribute:

- `all_exist` – Every item being collected exists on the target system
- `any_exist` – Any number of items (zero or more) collected exist on the target system

- at_least_one_exists – At least one item (1 or more) must exist on the target system
- none_exist – Zero items may exist on the target system
- only_one_exists – One and only one item must exist on the target system

Note the value for the “object_ref” attribute. This needs to be unique identifier for the <rpminfo_object> being collected from the target system.

Inside the <objects> section, create a <rpminfo_object> using the Linux namespace.

```
<rpminfo_object
   id="oval:org.cisecurity.oval:obj:99"
   xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux"
   version="1">

    <name>RPM_Package_Name</name>
</rpminfo_object>
```

Note that the <name> element is mandatory and specifies which RPM package to collect from the target system.