Creating an OVAL Report Template

The OVAL Reports Schema is a new category of the OVAL Schema. Its purpose is to drive collection of system state information and format it into a report. This report can then be used as evidence of compliance or non-compliance with a given configuration recommendation or policy. This document provides an overview of the OVAL Reports Schema and a quick tutorial as to how to use this schema to create an OVAL Report Template file.

OVAL Overview

Open Vulnerability and Assessment Language (OVAL™) is an international, information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services. OVAL includes a schema used to encode system details, and an assortment of content repositories held throughout the community. The schema standardizes the three main steps of the assessment process: representing configuration information of systems for testing; analyzing the system for the presence of the specified machine state (vulnerability, configuration, patch state, etc.); and reporting the results of this assessment. The repositories are collections of publicly available and open content that utilize the schema.

The OVAL community has previously developed four schemas to serve as the framework and vocabulary of the OVAL Language. These schemas support three steps of the assessment process: an OVAL System Characteristics schema for representing system information, an OVAL Definition schema for expressing a specific machine state, an OVAL Variables schema to support flexibility in definitions, and an OVAL Results schema for reporting the results of an assessment. OVAL Reports represent a fifth XML schema within OVAL.

OVAL Reports Overview

The OVAL Reports Schema is used by an author to identify system information to collect and to then format that information into a structured document. Unlike an OVAL Definition, which drives the automated assertion of system state, an OVAL Report Template seeks to drive the presentation of discovered information, making no judgment about correctness, giving users an awareness of important characteristics of a system's configuration. If an evaluation based on the generated report is desired, other tools (such as the Open Checklist Interactive Language, or OCIL) will need to be employed.

An OVAL Report Template file, which is an XML file that follows the OVAL Reports Schema, consists of three or four parts. As with all OVAL files, the first part consists of a generator element which identifies when a file was created, the schema version, and optionally additional information, such as the name and version of tools used to assist the creation. Following this is the oval_definitions section that defines the system objects that need to be collected from the system. This section reuses the Object structure found in an OVAL Definition and is identical to a file created using the OVAL Definition schema except that it does not contain the Definition or Test blocks, as such structures are unnecessary in an OVAL Report Template. The third section

is an XSLT 1.0 stylesheet. The stylesheet contains all the instructions used to organize and format the output report and can be understood by any XSLT 1.0 processor. Finally, the template file may contain an optional signature element using XMLDSig. This can be included to provide some assurance of the integrity of the file.

In addition to its XML schema, OVAL Report also includes a set of XSLT templates designed to handle common formatting tasks. These templates exist in a separate XSLT file and thus can be used independently of any OVAL Report Template file. An overview of the templates in this file is provided below.

Motivations and Use Cases

The OVAL Reports Schema was developed in response to a community need to support system configuration recommendations for which it was possible to automatically collect the relevant information, but where it was not possible to evaluate this information in an automated way. A prime example of this would be recommendations that deal with access rights as they often are expressed as "Limit access to a given resource only to trusted individuals who need access for their jobs." While it is often possible to automatically extract a list of people who have access to a resource, the list of users who belong on that list will vary from location to location. With OVAL Reports, a report could be generated listing the users of interest, possibly cross-correlated with other relevant characteristics of those users, and the person performing the assessment could then use the report to determine compliance with the given recommendation.

As noted, one of the primary use cases of OVAL Reports is to create a report that can be used to determine compliance with a system configuration recommendation. However, OVAL Reports do not contain any structures to query a reader as to whether the output report document indicates adherence to a given recommendation. There is an existing schema for the creation of an interactive questionnaire in the form of OCIL. This can be accomplished by utilizing an XCCDF complex-check where the first component check generates the OVAL Report output and the second component check is an OCIL check that prompts the user to review the report and answer a questionnaire. For more information about OCIL and XCCDF please see http://scap.nist.gov/specifications/ocil/index.html and http://scap.nist.gov/specifications/xccdf/, respectively.

Creating an OVAL Report Template

An OVAL Report Template consists of four blocks: the generator, the oval_definitions, and the stylesheet are all required blocks while the signature block is optional. This section will go through each of these blocks, describe its purpose, and provide pointers regarding its construction.

<generator>

An OVAL Report Template begins with a generator block. The generator element uses the GeneratorType defined in the OVAL Common Schema and utilized by many other OVAL schemas, including the Definitions, System Characteristics, and Results schemas. The generator block must identify the OVAL schema version and must also include a timestamp. It may

optionally include additional information such the name and version of the editor used to generate the OVAL Report Template file.

<oval_definitions>

Following the generator block is the oval_definitions block. The structure of this element is identical to that of an OVAL Definition file. However, note that both the <definitions> and <tests> elements must not appear in an OVAL Report Template. This allows authors to utilize existing OVAL Content when creating a Report Template file with minimal modifications. The generator block within the oval_definitions is still required and holds generator information associated with the oval_definitions block, primarily for situations where existing OVAL Content is being re-used. If the content of the oval_definitions block was created fresh rather than copied from pre-existing content, the oval_definitions generator would be identical to the generator at the beginning of the Report Template file.

Creating an OVAL Report Template file follows most of the steps one would use when creating an OVAL Definition file. The primary structures of interest are OVAL Objects since it is these elements that identify the system data that will be formatted into a report. OVAL States are also used for their ability to filter OVAL Objects using the Object "set" structures. As mentioned before, OVAL Reports do not include any evaluation of discovered state so OVAL States are not used in this capacity. Similarly, since OVAL Definition and Test structures exist primarily to structure the evaluation of machine state, neither of these are relevant in an OVAL Report Template and they should not appear in the oval_definitions block. OVAL Variables fulfill the same uses in OVAL Reports as they do in OVAL Definition files. Finally, the oval_definitions section may contain its own set of XMLDSig signatures so re-used definition structures can retain their original integrity checks.

For each type of included structure (Objects, States, Variables, and signatures), OVAL Reports use the same schemas and structures used when creating an OVAL Definitions file. Readers are directed towards the document on the OVAL web site (http://oval.mitre.org/oval/about/documents.html) for more guidance on creating OVAL Definition files.

<stylesheet>

The stylesheet section is used to format the final report file from the raw information collected by OVAL. The stylesheet block is an XSLT stylesheet and its contents follow the structures of the XSLT 1.0 language. (The XSLT specification is available from the W3C: http://www.w3.org/TR/xslt.)

The stylesheet section uses the OVAL System Characteristics file that would be produced in response to the collection of the OVAL Objects defined in the oval_definitions section. Authors create XSLT instructions to process this source file into the desired output report. All the XML processing abilities of XSLT are available to authors for the processing and presentation of the desired information. In addition, OVAL Reports include a suite of helper templates that perform commonly needed processing actions:

- getOvalItemsOfOvalObject – Takes the ID string of an OVAL Object and returns all the System Characteristic Items that were collected on behalf of the named Object.

- getOvalObjectDiscoveryStatus – Takes the ID string of an OVAL Object and returns the string of that Object's flag attribute, which indicates whether the Object's target system state was successfully collected.
- getOvalObjectOfOvalItem – Given the ID string of a System Characteristics Item, return the ID string of the OVAL Object for which it was collected.
- getVarValsFromOvalObject – Given the ID strings of an OVAL Object and an OVAL Variable, return the value of the named Variable at the time it was used by the named Object as a string.

The list of templates continues to evolve based on the needs of the OVAL Community and is likely to grow in the future. All these templates exist in a separate XSLT file and, as such, could be imported and used in any XSLT 1.0 document.

<Signature>

This optional section of the OVAL Report Template allows authors to include integrity checks for the document as a whole. The body of this element follows the XMLDSig standard. (The XMLDSig specification is available from the W3C: http://www.w3.org/TR/xmldsig-core/.)

Example Report Template

This section provides a simple example of an OVAL Report Template along with the output it might produce on some system. The OVAL Report Template in this example collects the list of groups and users listed in the ACL of a particular file (regedit.exe) and displays those users in a table along with their effective permissions to the file. Because the permissions themselves are associated with SIDs and SIDs are not easily understood by human readers, the SID associated with each set of permissions is translated into a human-readable name whenever possible.

Figure 1. An OVAL Report Template

```
1  <oval_report_template xmlns="http://oval.mitre.org/XMLSchema/oval-reports-5"
                         xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                         xsi:schemaLocation="http://oval.mitre.org/XMLSchema/oval-reports-5
                                             oval-reports-schema.xsd
                                             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">
2     <generator>
3        <oval:schema_version>5.8</oval:schema_version>
4        <oval:timestamp>2010-04-09T11:13:00-06:00</oval:timestamp>
5     </generator>
6     <oval_definitions xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"
                        xmlns:win-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows">
7        <generator>
8           <oval:schema_version>5.6</oval:schema_version>
9           <oval:timestamp>2009-12-17T10:41:00-05:00</oval:timestamp>
10       </generator>
11       <objects>
12          <fileeffectiverights53_object
                           xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
                           id="oval:example:obj:1" version="1"
                           comment="Collect the permissions of all of the SIDs who have
                                    permissions that are explicitly granted or denied in the
                                    ACL of the file 'C:\WINDOWS\regedit.exe'." >
```

```
13            <path>C:\WINDOWS</path>
14            <filename>regedit.exe</filename>
15            <trustee_sid operation="pattern match">.*</trustee_sid>
16         </fileeffectiverights53_object>
17         <sid_sid_object xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#windows"
                           id="oval:example:obj:2" version="1"
                           comment="Collect the trustee_name (username) and trustee_domain
                                    information for the SIDs listed in the ACL of the file
                                    'C:\WINDOWS\regedit.exe'.  This object will serve as a
                                    mapping between the SIDs in the ACL and their human-
                                    readable usernames." >
18            <trustee_sid var_ref="oval:example:var:1" var_check="at least one"/>
19         </sid_sid_object>
20      </objects>
21      <variables>
22         <local_variable id="oval:example:var:1" version="1" datatype="string"
                           comment="This variable represents all of the SIDs, on the system,
                                    that have permissions explicitly granted or denied in
                                    the ACL of the file 'C:\WINDOWS\regedit.exe'.">
23            <object_component object_ref="oval:example:obj:1" item_field="trustee_sid"/>
24         </local_variable>
25      </variables>
26   </oval_definitions>
27   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                     xmlns:win-sc=
                           "http://oval.mitre.org/XMLSchema/oval-system-characteristics-5#windows"
                     version="1.0">
28      <xsl:import href="oval-report-templates.xsl"/>
29      <xsl:template match="/">
30         <xsl:variable name="perms">
31            <xsl:call-template name="getOvalItemsOfOvalObject">
32               <xsl:with-param name="objid">oval:example:obj:1</xsl:with-param>
33            </xsl:call-template>
34         </xsl:variable>
35         <xsl:variable name="users">
36            <xsl:call-template name="getOvalItemsOfOvalObject">
37               <xsl:with-param name="objid">oval:example:obj:2</xsl:with-param>
38            </xsl:call-template>
39         </xsl:variable>
40         <html>
41            <head><title>User access to regedit.exe</title></head>
42            <body>
43               <xsl:text>The following users are named in the ACL for regedit.exe</xsl:text>
44               <br/>
45               <table border="true">
46                  <tr><th>User (SID)</th><th>Granted permission</th></tr>
47                  <xsl:for-each select="$perms/*">
48                     <tr>
49                        <td>
50                           <xsl:variable name="current-user-sid">
51                              <xsl:value-of select="./win-sc:trustee_sid"/>
52                           </xsl:variable>
53                           <xsl:value-of select="$users/*[./win-sc:trustee_sid =
                                                          $current-user-sid]/win-sc:trustee_name"/>
54                           <xsl:text> (</xsl:text>
55                           <xsl:value-of select="$current-user-sid"/>
56                           <xsl:text>)</xsl:text>
57                        </td>
58                        <td>
59                           <xsl:variable name="permission-list">
```

```
60                                 <xsl:for-each select="node()[.='1']">
61                                    <xsl:value-of select="local-name()"/>
62                                    <xsl:text> </xsl:text>
63                                 </xsl:for-each>
64                              </xsl:variable>
65                              <xsl:choose>
66                                 <xsl:when test="$permission-list=''">NO ACCESS</xsl:when>
67                                 <xsl:otherwise>
68                                    <xsl:value-of select="$permission-list"/>
69                                 </xsl:otherwise>
70                              </xsl:choose>
71                           </td>
72                        </tr>
73                     </xsl:for-each>
74                  </table>
75               </body>
76            </html>
77         </xsl:template>
78      </xsl:stylesheet>
79</oval_report_template>
```

We can recognize several significant segments in the above template:

- Lines 2-5 – This is the generator block of the Report Template itself
- Lines 6-26 – This is the oval_definitions block
    - Line 7-10 – This is the generator of the oval_definitions block. Note that this differs from the generator block of the Template itself indicating that the definition content came from another source.
    - Line 12-16 – This defines a fileeffectiverights53_object. This collects all the permissions explicitly listed in the ACL of the named file (regedit.exe).
    - Line 17-19 – This defines the sid_sid_object. Sid_sid_objects collect the user and domain information associated with a given SID. We use the information discovered by this Object to effect a translation from SID to human-readable usernames.
- Lines 27-78 – This is the stylesheet block
    - Line 28 – This imports the oval-report-templates stylesheet, which defines the helper templates that come with OVAL Reports. We need to import this stylesheet to make use the templates.
    - Line 29 – This starts the template definition. Notice that we match the "/" pattern to ensure this template will be applied to the whole system-characteristics file. Using more targeted matches is also possible, although most of the helper templates automatically operate off of the root of the system-characteristics file.
    - Line 30-39 – We populate two variables, one for each of the OVAL Objects we collected. We use the getOvalItemsOfOvalObject template from the helper templates stylesheet to populate these variables with the system-characteristic Items associated with each of the OVAL Objects.
    - Line 40-46 – These start the HTML document, set up the document header, and set up a table in the body of the document.
    - Line 47-48 – We use the XSLT for-each element to iterate through each set of permissions retrieved by the fileeffectiverights53_object, creating a new table row for each permission set.

- Line 50-56 – In the first cell of the row we create a new variable that contains the SID as it appears in the current permission Item. We then print out the name collected from the sid_sid_object that matches the SID variable and then add the SID itself in parenthesis
- Line 59-70 – In the second cell of the row we start by creating a new variable that contains the name of all the elements in the permission Item whose values are 1. These correspond to the permissions that have been granted. If the permission variable is empty we write "NO ACCESS", indicating that the given SID is explicitly given no access to the file. Otherwise, we write out the contents of the variable, thus enumerating all permissions granted to that SID in the file's ACL.
- The remaining lines of the Report Template close out the table, the HTML document, the stylesheet, and the Report Template itself.

The following table shows the start of the output report produced by the above Report Template when run against one particular system.

Figure 2. OVAL Report Output

The following users are named in the ACL for regedit.exe

User (SID)	Granted permission
SYSTEM (S-1-5-18)	standard_delete standard_read_control standard_write_dac standard_write_owner standard_synchronize generic_read generic_write generic_execute generic_all file_read_data file_write_data file_append_data file_read_ea file_write_ea file_execute file_delete_child file_read_attributes file_write_attributes
OVALTEST-PC\IUSR_OVALTEST-PC (S-1-5-21-1568420705-1441692396-607190668-1028)	NO ACCESS
Administrators (S-1-5-32-544)	standard_delete standard_read_control standard_write_dac standard_write_owner standard_synchronize generic_read generic_write generic_execute generic_all file_read_data file_write_data file_append_data file_read_ea file_write_ea file_execute file_delete_child file_read_attributes file_write_attributes
Users (S-1-5-32-545)	standard_read_control standard_synchronize generic_read generic_execute generic_all file_read_data file_read_ea file_execute file_read_attributes
...	...

This is a very simple sample and one could easily envision more elaborate OVAL Report Templates that could do things like convert groups into actual lists of users or correlate with other system information. Authors can make use of the full capabilities of XSLT 1.0 as well as all the system information one can specify using OVAL. Because of this the OVAL Reports

Schema offers a great deal of power and flexibility to authors for the purpose of documenting system configuration information.

Conclusion

This document has provided an introduction to the OVAL Reports Schema. This is a living document since the OVAL Reports Schema will continue to evolve to meet the needs of the OVAL Community. Towards that end, all feedback and suggestions are welcomed – please send comments to the OVAL Developer List (oval-developer-list@lists.mitre.org) or, if you wish your comments to remain private, send them to the OVAL Moderator directly (oval@mitre.org). Please help us keep this schema useful and responsive to the needs of the security automation community.