OVAL™ Language Design Document

Version 5.1

Introduction

The purpose of this document is to present a detailed discussion of the design of the OVAL Language. It also serves as an introduction to those interested in gaining a greater understanding of the language, how it can be used, and how it can be incorporated into a security application.

The language requirements are specified within the *OVAL Language Requirements* document, and serve as the basis for the design presented within this document. Throughout this document, specific requirements are called when those requirements are addressed by implementation details being discussed.

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 language used to encode system details, and an assortment of content repositories held throughout the community. The language 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 language.

The OVAL community has developed three schemas written in Extensible Markup Language (XML) to serve as the framework and vocabulary of the OVAL Language. These schemas correspond to the 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, and an OVAL Results schema for reporting the results of an assessment.

Content written in the OVAL Language is located in one of the many repositories found within the community. One such repository, known as the OVAL Repository, is hosted by The MITRE Corporation. It is the central meeting place for the OVAL Community to discuss, analyze, store, and disseminate OVAL Definitions. Each definition in the OVAL Repository determines whether a specified software vulnerability, configuration issue, program, or patch is present on a system.

The information security community contributes to the development of OVAL by participating in the creation of the OVAL Language on the OVAL Developers Forum and by writing definitions for the OVAL Repository through the OVAL Community Forum. An OVAL Board consisting of representatives from a broad spectrum of industry, academia, and government organizations from around the world oversees and approves the OVAL Language and monitors the posting of the definitions hosted on the OVAL Web site. This means that the OVAL, which is funded by US-CERT at the U.S. Department of Homeland Security for the benefit of the community, reflects the insights and combined expertise of the broadest possible collection of security and system administration professionals worldwide.

Implementation of the Language

The OVAL Language has been created to facilitate the exchange of information between security vendors and tools. The language is composed of three XML schemas; the OVAL Definition schema expresses a specific machine state, the OVAL System Characteristics schema stores configuration information gathered from a system, and the OVAL Results schema presents the output from a comparison of an OVAL Definition against an OVAL System Characteristics instance.

OVAL Definition Schema

The OVAL Definition schema provides the framework for constructing logical statements of the following form:

> Is operating system X installed?
> AND Is service Y enabled?
> AND Is user Z allowed to use service Y?

Each of these questions is represented by a specific machine configuration parameter, and an associated value. The collection of these questions results in an expression of a desired machine state, which can be applied to a specific system to determine its configuration in comparison.

At the highest level, and OVAL Definition document consists of six distinct sections; generator, definitions, tests, objects, states, and variables. Each document typically consists of one or more individual OVAL Definitions, where the information in the tests, objects, states and variables sections can be shared across multiple definitions *(Req. 2.2)*. The entire document can be signed using an optional digital signature (enveloped) which is based upon the W3C XML Signature standard (http://www.w3.org/Signature/).*(Req. 1.4)*

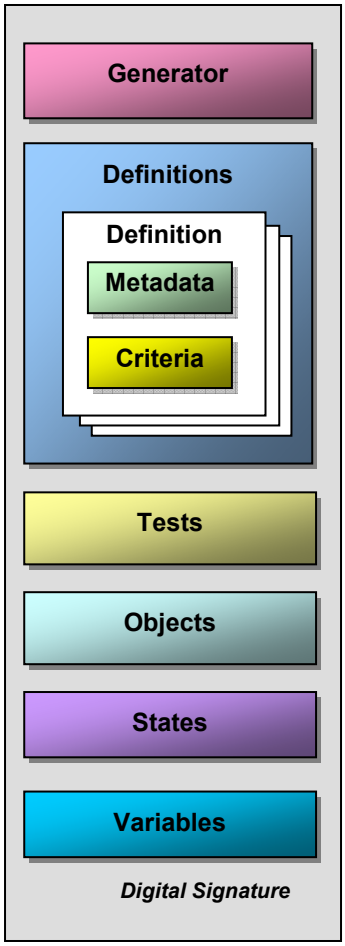

Figure 1 : OVAL Definition Document

<generator>

The generator section is provided as an information resource to tools consuming the definition document. The purpose of this section is to allow consumers to determine the tool that was used to generate the document *(Req. 2.8)*, the schema version to which the document applies *(Req. 2.6)*, and a timestamp for when the document was generated *(Req. 2.7)*.

Note that the timestamp element does not specify when a definition (or set of definitions) was created or modified but rather when the actual XML document containing the definition(s) was created.

<definitions>

The definitions section serves as the container for each of the definitions contained within an instance document. Each definition is distinguished by a globally unique identifier of the form "*oval:Organization DNS Name:ID Type:ID Value*" where *(Reqs. 2.11, 2.13)*:

- Organization DNS Name is of the form 'org.mitre.oval', and not only provides a level of uniqueness to the ID, but also provides a source for questions and information about a specific definition.
- ID Type denotes the entity to which the ID is being applied *(Req. 2.12)*, and can be one of the following values:
    - def – Definition
    - obj – Object
    - ste – State
    - tst – Test
    - var – Variable
- ID Value is an integer that is unique to the DNS name and ID Type pair that precedes it.

Note, the ID format is not specific to the definition ID, but extends across all of the globally reusable components in the language - definitions, objects, states, tests, and variables.

Each definition can be further decomposed into a <metadata> and a <criteria> section. The <metadata> section provides descriptive header information associated with the definition, including; short title, affected platform information, definition description, and references to more information *(Req. 2.14)*. It is important to note that the affected platform information is provided solely for informational purposes and is not meant to be used when examining the state of a system. For example, the affected platform section may contain the string 'Windows XP', but the actual OVAL statement to test for the existence of Windows XP on a system is contained elsewhere in the definition.

It has also been recognized that organizations may have meta-data requirements that are either, too specific to become part of the official OVAL Language, or can not be added to the schema in a timely enough manner. For these reasons, an <xsd:any> element has been included as part of the <metadata> section. The purpose for this element is that it allows an area of a schema – in this case the <metadata> section – to be extended, such that any information can be included at the <xsd:any> directive, but it is not considered during validation of the document against the schema.

The <criteria> section provides the means for composing a logical combination and nesting of tests *(Req. 2.3)* and/or other definitions *(Req. 2.1)*. The resultant logical expression is the actual definition statement for the issue in question. In practice, the criteria is composed of a collection of individual criterion statements, and nested criteria blocks, which are bound by the logic operators AND, OR, and XOR. Each criterion statement contains a reference to an individual system level test, a negation flag for controlling the interpretation of a test's TRUE/FALSE result *(Req. 2.4)*, and a comment describing the intent of the test.

In addition to the <metadata> and <criteria> sections, an optional <notes> section can be included within the definition. The purpose of the notes section is to provide more detailed documentation about specific technical aspects of the definition.

Each individual definition can be digitally signed using an XML signature. Note that it is just the <definition> element (and all its children) being signed in this case and not the referenced tests, objects, and states.

<tests>

The tests section contains all of the individual system level tests needed by the definitions *(Req. 2.10)*. For each platform or application addressed by OVAL, there is a corresponding set of tests to express the various configuration states for that entity *(Req. 2.15)*. The structure of each test, with respect to the information being collected, is meant to reflect the structure of the information for the corresponding entity on the end host *(Req. 2.16)*. For example, for the Microsoft Windows platform, there are tests for Active Directory, the Registry, and file data. For Red Hat Linux, there are tests for RPM, uname, and file data. The tests are intended to be specific to the entity being tested, so even though there is are 'file data' tests for both Microsoft Windows, and Red Hat Linux, the nature of each of these tests is different, because the representation of a file in each environment is different.

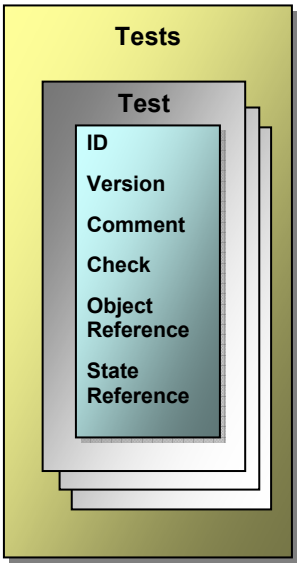

Figure 2 : OVAL Definition Test Structure

While the intent of each individual test may be very different, the structure of each test is actually very similar. A test statement includes a unique id, a version, a comment, a check, an object reference, and a state reference. The test ID, the format of which is described in the <definitions> section above, is referenced in the criteria section within a definition to build the definition statement. The test version is provided to enable tracking and management of modifications to a test over its lifetime. The test comment provides a brief description of the purpose of the test.

The check value is used to define the relationship between the objects returned from the Object Reference, and the state returned from the State Reference, when analysis is performed. The allowable values for the check attribute are 'all', 'at least one', 'only one', and 'none exists'. The meaning of these values with respect to the object/state relationship is as follows:

- all – All of the identified objects must match the state

- at least one – At least one of the identified objects must match the state
- only one – One and only one object must match the state
- none exists – None of the objects can match the state

The final two components within a test are references to an object and a state. The binding of these two entities represents the specific, end system, configuration state that is being tested. A complete definition of objects and states is presented later in this document.

Each individual test can be digitally signed using an XML signature. Note that it is just the <xxxx_test> element (and all its children) being signed in this case and not the referenced objects, and states.

<objects>

The objects section contains all of the system entities (e.g. files, Registry keys, packages…) that are being analyzed by a definition *(Req. 2.10)*. For each object of interest on the platforms supported by the OVAL Language, there is a corresponding schema detailing the properties being examined for that object. For example, in the case of a Microsoft Windows Registry object, the properties being examined are the hive, key, and name, which represent the path components to a specific Registry key value. The reason for dedicating a specific section of the definition to objects is, in many cases, the same objects are being examined over and over again. By assigning unique ID's to the objects and maintaining them in a separate section, it allows the same objects to be referenced from multiple tests, and eases the task of data collection.

For some objects, it may be desirable to control the way in which it is gathered, or the amount of data gathered. For this reason, <behavior> and <filter> elements provide a means to perform such actions on an object. A <behavior> is meant to control the way in which data is gathered for an object. For example, in the case of an object that represents a directory, a <behavior> would state whether to recurse the directory, and in which direction the recursion should occur. In the case of a <filter>, it is used to bound, or restrict the number of matching objects collected. Each filter is used to remove a specific subset of objects from a defined object set. For example, an OVAL Object might be defined to represent all of the files in a particular directory. A filter may then be applied to remove all of the files from this set whose size is greater than 100K.

Each individual object can be digitally signed using an XML signature.

<states>

The states section complements the objects section, in that it contains the values against which the actual system objects are being compared *(Req. 2.10)*. Returning to the Microsoft Windows Registry example from the object section – the object defines the Registry path to traverse, while the corresponding state defines the specific value of interest, and the operation to be performs when conducting a comparison. As explained in the object section, it is possible to reference a single object from multiple tests. It is also possible that the value comparison being conducted against a specific object can differ from one test to the next. Therefore, the states are defined in a distinct section, and each has an associated unique ID to allow for greater flexibility in the reuse and combination of objects and states within a test.

Each individual state can be digitally signed using an XML signature.

<variables>

It is not always the case that all of the values within a definition are known at the time a definition is written *(Req. 2.5, 2.10)*. Some values can vary based upon an organization's policy or the specific configuration of a machine. As a result, the variables section contains placeholders for values that must be filled in at definition execution time – with one exception. There are three types of variables defined in OVAL, external, local, and constant. External variables are values that have to be obtained from an outside source – e.g. user, configuration file… - and plugged into the definition at run time. Local variables are also introduced into the definition at run time, but they represent values that can be retrieved from objects on the system itself. For example, an environment variable holds the file path to an application, and the definition needs that value in order to build the full path to one of the application's files. The value of the local variable obtained from the environment variable can differ from one system to the next, depending

upon how the application was installed. Finally, constant variables represent common static values used across a number of definitions. These variables have been broken out and hard coded into the individual definition document to ensure consistency across definitions.

One final interesting aspect of variables is that it may be necessary to change the value of an individual variable during a single analysis of a definition document. For example, a variable may be reused within multiple definitions, but its value may vary from one definition to the next, based upon the context of the definition itself. In these cases, when the OVAL System Characteristics, and OVAL Results documents are created, an instance attribute is associated with each variable to ensure that the correct value is associated with the correct definition.

Each individual variable can be digitally signed using an XML signature.

OVAL System Characteristics Schema

The OVAL System Characteristics schema defines a standard XML format for storing system configuration information. This configuration information includes OS parameters, installed software application settings, and other security relevant configuration values. The purpose of this schema is to provide a "database" of system characteristics against which the OVAL Definitions can be compared in order to analyze a system for a defined machine state. In essence, the schema defines a standard system characteristics exchange format that can be incorporated into a variety of tools, such as those listed in the OVAL-Compatible Products and Services section on the OVAL Web site.

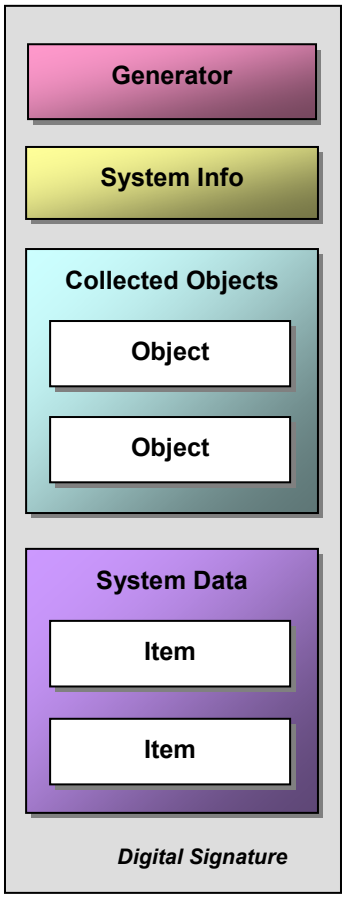

Figure 3 : OVAL System Characteristics Document

At the highest level, and OVAL System Characteristics document consists of four distinct sections; generator, system_info, collected_objects, and system_data. Each instance document consists of the configuration data collected from a single system. Enveloping the entire document is an optional digital signature which is based upon the W3C XML Signature standard (http://www.w3.org/Signature/) *(Req. 1.4)*.

<generator>

The generator section is provided as an information resource to tools consuming the system characteristics document. The purpose of this section is to enable consumers to determine what tool was used to generate the document *(Req. 3.4)*, the schema version to which the document applies *(Req. 3.2)*, and a timestamp for when the document was generated *(Req. 3.3)*.

<system_info>

The system_info section contains the details about the installed operating system, system architecture, and network interfaces. The primary reason for gathering this information is to provide a binding between a specific system and its corresponding system characteristics file *(Req. 3.5)*.

<collected_objects>

The collected_objects section provides a mapping between the objects contained within a specific Definition document and the corresponding items gathered from the system *(Req. 3.7)*. Each collected object element maintains the object id from the Definition file, a reference pointer to the associated item(s) within the system_data section, and a flag which conveys the result of locating and gathering the object from the system *(Req. 3.6)*. The set of flags are:

- complete – Every instance of the object was collected and stored in the system characteristics file.
- does not exist – The object could not be located on the system.
- error – An error occurred while collecting the object information

- incomplete – Only a subset of the matching objects on the system were collected.
- not collected – No attempt was made to gather the object information from the system.
- not applicable – The specified object is not applicable to the system being characterized.

Note that existence of the object on the system can not be inferred from the 'incomplete' and 'not collected' flags.

Due to the fact that objects in the definition can be represented as pattern matches, and that multiple items on the system can match the pattern, it is possible for a collected object to have the reference multiple items. For example, a Windows File Object might use the following pattern match, 'C:\Windows*.dll'. This will result in a system characteristics document with an individual item for each DLL file in the C:\Windows directory, with each item having a unique reference identifier. The definition is expected to contain the necessary instructions for how to handle multiple items contained within the system characteristics document.

<system_data>

The system_data section contains information about the actual items gathered from the system. For each item in this section, there is a unique reference identifier, and a status flag to report the result of the data collection. A single item may be referenced in more than one collected object section.

OVAL Results Schema

The OVAL Results schema defines a standard XML format for storing the results of an OVAL evaluation of a system. The results data contains the current state of a system's configuration as compared against a set of OVAL Definitions. The OVAL Results Schema allows applications to consume this data, interpret it, and take the necessary actions to mitigate the vulnerabilities and configuration issues (for example, install patches, alter system configuration settings, and/or take external precautions to limit access to the affected systems). This schema also defines a standard vulnerability and misconfiguration exchange format that can be incorporated into a variety of tools, such as those listed in the OVAL-Compatible Products and Services section on the OVAL Web site.

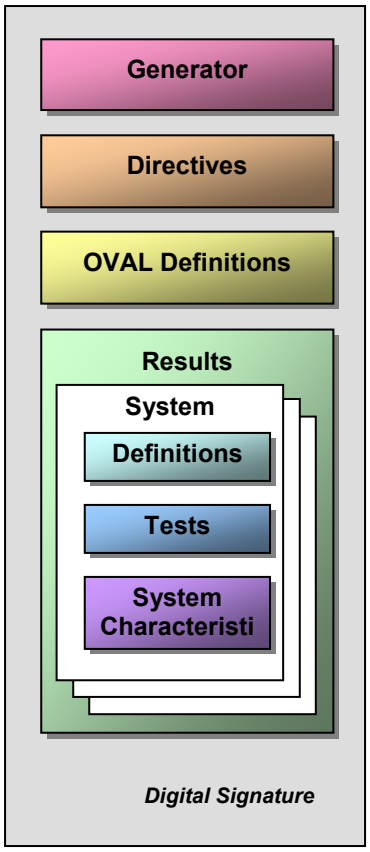

Figure 4 : OVAL Results Document

At the highest level, an OVAL Results document consists of four distinct sections; generator, directives, OVAL Definitions, and results. Each instance document can consist of results from multiple systems, analyzed against the same set of OVAL Definitions. Enveloping the entire document is an optional digital signature which is based upon the W3C XML Signature standard (http://www.w3.org/Signature/) *(Req. 1.4)*.

<generator>

The generator section is provided as an information resource to tools consuming the results document. The purpose of this section is to enable consumers to determine what tool was used to generate the document *(Req. 4.4)*, the schema version to which the document applies *(Req. 4.2)*, and a timestamp for when the document was generated *(Req. 4.3)*.

<directives>

An OVAL Result document contains a lot of data, much of which is not always needed by those consuming it. In order to accommodate the needs of all of the various results users, many of the elements within the schema are configured as optional. The role of the directives is to allow the Results Producer to indicate which elements are included in the document and which are not *(Req. 4.1)*. For the Results Consumer, the directives allow for a quick determination as to whether the contents of the document fulfill its needs or not. The directives defined in this section are:

- definition_true – Report definitions whose result evaluated to true.
- definition_false – Report definitions whose result evaluated to false.
- definition_unknown – Report definitions whose result evaluated to unknown.
- definition_error – Report definitions whose result evaluated to error.
- definition_not_evaluated – Report definitions that were not evaluated.
- definition_not_applicable – Report definitions that do not apply to the system in question.

With each directive, there are associated two attributes:

- reported (true/false) – State of associated directive within the document. In effect, this attribute serves as a flag to indicate whether the information associated with a given directive is contained within a result document.

- content (thin/full) – Depth with which associated directive results are reported.

    - A value of 'thin' means only a minimal amount of information is provided. This is the id associated with an evaluated OVAL Definition and the result of the evaluation. The criteria child element of a definition is not present when providing thin results. In addition, system characteristic information for the objects used by the given definition is also not present.

    - A value of 'full' means that very detailed information is provided allowing in-depth reports to be generated from the results. In addition to the results of the evaluated definition, the results of all extended definitions and tests included in the criteria, as well as the actual information collected from the system, is present.

<oval_definitions>

The oval_definitions section is in fact the actual set of OVAL Definitions that were analyzed in order to generate the results document *(Req. 4.4.2)*. While including this document within the results certainly expands the size of the document, it eliminates the need to always have the definitions document present when analyzing the results document. It is believed that the convenience and reliability gained by including this information within the results, outweighs the impact of the larger document. Inclusion of the definitions document within the results document is optional.

<results>

The results section serves as a container for the analysis results of potentially numerous systems against the OVAL Definition document included in the oval_definitions section. The results section is composed of a number of system elements. Within each system element are the following sub-elements:

- definitions – The analysis results for each OVAL Definition against a specific system *(Req. 4.6)*.

- tests – The analysis results for each test contained within the OVAL Definitions against a specific system *(Req. 4.7)*.

- oval_system_characteristics – The oval_system_characteristics section is a copy of the OVAL System Characteristics document for the system, which was generated as a result of analyzing the given OVAL Definitions document *(Reqs. 4.5)*. As with the oval_definitions section above, this information is provided as a convenience to ensure that the results consumer has access to all of the information that was available to the results producer. Unlike the oval_definitions section, it is required to have the System Characteristics document present.

Additional Information

For a more detailed definition of each of the OVAL Language elements, please consult the element dictionaries or refer to the documentation contained within the individual schema documents. For these documents and additional

information about OVAL, please visit to the OVAL Web site at http://oval.mitre.org or send an email to oval@mitre.org.