OVAL™ Language Requirements

Version 5.1

Introduction

IT Security Professionals around the world all face a common dilemma when it comes to securing their systems – a way to determine if a system is truly vulnerable or out of compliance with the stated policy. For example, when a security advisory is released for a specific issue it is normally a text document and often doesn't contain all of the information necessary to determine if the vulnerability exists on a specific system or not. The IT Professional is left with the task of investigating all of the available sources for this issue and then trying to piece together the details for detecting this issue.

Content teams for vulnerability scanning and/or remediation tools face a similar problem in that they are forced to reverse-engineer security advisories to develop tests for their tools. The result is that, based upon the interpretation of the individual analysts, different approaches are taken for different tools when searching for the existence of a vulnerability. As a consequence, executing different tools on a specific system can result in conflicting answers. Again, the burden falls upon the IT Professional to deconflict the results by examining the individual approaches and, if possible, decide which is correct.

This problem isn't limited to just vulnerabilities, but to any case where an organization is publishing security configuration information about computer systems. There are often multiple repositories of configuration information, multiple ways in which to manipulate that data, and in some cases, complex precedence relationships between the data. It is time-consuming and error-prone for an individual to read a document, interpret its meaning with respect to a specific configuration setting, and then apply that knowledge to an actual system to determine an answer.

What the industry requires is a standardized way for expressing the configuration state of computer system. The presentation of this information should be in such a way as to easily facilitate its consumption by a software tool. The advantage of such a standard is that it would:

- greatly shorten the time between the official announcement of an issue and of tools being able to check for it;
- bring uniformity to the results produced by configuration scanning tools;
- assist in the exchange of information between security tools, and;
- reduce the need for IT Professionals to learn the proprietary languages of each of their tools, and instead allow them to learn a single language.

The purpose of this document is to present the Open Vulnerability and Assessment Language (http://oval.mitre.org) as a standard that fulfills these needs and requirements.

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.

OVAL Use Cases

Information security is a function that consumes significant resources from an organization, and is continually getting more difficult to manage. One of the biggest problems is the lack of standardization between the sources of security information, and the tools that consume that information, as well as between the various tools themselves. Often, the exchange of security information is time critical, but is hampered by the variety of incompatible formats in which it is represented. The cases below exemplify some of the need for a standard like OVAL, and its potential uses in the security industry.

Security Advisory Distribution

One acknowledged need within the security industry is for application and operating system vendors to publish vulnerability information in a standard, machine-readable format. The benefit of this is two-fold: first, it provides scanning tools with immediate access to content that can be used to assess the security posture of a system, and second, it moves the authoring of the technical details of a vulnerability from the reverse-engineering efforts of the scanner tool developer, to a more authoritative source, the developer of the vulnerable software.

Vulnerability Assessment

Currently, organizations that develop vulnerability assessment tools also need to employ a team of content developers. The role of this team is to investigate vulnerabilities as they become known, gather all of the available information for a given vulnerability, run various tests against live systems to examine the parameters that indicate the presence of a vulnerability, develop the tests in the language of the tool in question, and do this all under a very strict time requirement. The final requirement obviously runs counter to those before it, and results in the potential for analysis to not be completely fulfilled before a test absolutely has to be disseminated to the vendor's customers.

For vendors, having vulnerability information structured in a standard format allows them to quickly consume data from multiple sources, and lets them refocus some of their resources away from content generation, and onto tasks to enhance the functionality of their tool.

From the tool customer's perspective, the primary requirement for having a standard content format is that it demystifies the vulnerability assessment process and provides them with the ability to do apples-to-apples comparisons of tools. Having a well-documented, standard format provides users with the information they need to be able to understand the details of an issue, and to determine how a specific tool is conducting its business. The open standard also provides users with the opportunity to generate their own statements in the language and, if a tool allows, interpret them. When conducting tool comparisons, given a specific set of definitions, each tool tested should return the same result. If they do not, it would no longer be because the tools take different approaches to detecting a vulnerability. It would also remove the burden from the customer of determining which of the tools returns the most accurate results. The final outcome is that customers can now focus more on selecting a tool with the features that best meet their needs, and less on the more difficult problem of which of the tools does the correct job of detecting vulnerabilities.

Patch Management

The needs of the patch management vendors are similar to those of the vulnerability assessment vendors. There is some amount of reverse engineering that must be done in order to generate content for their tool to be able to apply a patch. Having this information generated by the software vendor and formatted in a standard format removes the reverse engineering requirement, and allows content to be provided to the end customers in a more timely fashion. A second requirement for this class of tools is to be able to easily consume vulnerability assessment results from a variety of scanning tools. If these results were offered in a standard format, interoperability between tools would no longer be an issue.

Configuration Management

Configuration management tools concern themselves with examining a machine's configuration state, comparing it against a known good or mandated configuration state, and reporting the results. There are a number of publicly available best practice configuration guides (e.g., The U.S. National Security Agency (NSA) Configuration Guides), and many more developed specifically for individual organizations. In many cases, these guides exist in paper form only and it is up to the IT Staff to translate the document into something that can be applied and enforced on a consistent basis. There are also automated solutions available, most notably the Center for Internet Security's Benchmark tools, which can scan a system for compliance against a given configuration and offer tailoring capabilities to suit the specific needs of an organization. Unfortunately, these tools often rely upon proprietary data formats, making it difficult to introduce new policies to the tool or to move data from one tool to another.

Having a standard language for expressing system configuration issues offers many benefits in this area. First, a single configuration specification need only be written once. It can then be consumed by any configuration management tool. Second, organizations can more easily develop and maintain their own configuration standards, as it only requires learning a single language and not a language specific to a particular tool. Finally, as with some of the cases above, divesting the language from the tool provides the tool vendor with a wider repository of content and allows them to focus more on functionality and features.

Auditing and Centralized Audit Validation

Audit validation is responsible for providing reports about the state of a machine at any given time in the past. There are two basic needs in this area; first is capturing machine configuration information at a level of granularity that allows an organization to monitor, track, and reconstruct the transition of a system's configuration from one state to another. The second need is that the data be stored in a standardized, data-centric format, thus ensuring that it is not bound to a specific tool, which may or may not be available at the time it is necessary to review the data.

Security Information Management Systems (SIMS)

Security Information Management Systems (SIMS) rely upon the output of a variety of security, auditing, and configuration tools, as well as their own agents, to build a comprehensive view of the security posture of an organization's network. Clearly, the fewer data formats that the SIM needs to understand the more flexible and powerful this tool can be. As with the Patch Management class of tools, standardizing the data exchange formats between tools greatly simplifies the interoperability requirements, and provides the end-users with a wider array of applications from which to choose.

System Inventory

A common issue that exists, not only for security tools, but any tool that is trying to conduct some sort of evaluation of a computer system, is determining the attributes of that system (e.g., operating system, patch level, installed applications, etc.), and being able to convey those attributes clearly and consistently. Currently, there is no universally accepted method for gathering the attributes from a system, nor is there a common way to express those attributes such that they can be easily consumed by another application. The need for this capability is widely acknowledged, and its use would be widespread.

Language Requirements

Based upon the goals of OVAL and the needs detailed in the use cases outlined above, the following set of requirements has been generated for establishing OVAL as the standard for expressing the configuration states of computer systems. At the highest level are the Basic Requirements, which capture the essence of the goals and use cases. Each of these requirements is further expanded and refined into individual classes of requirements in the OVAL Definition Requirements, OVAL System Characteristics Requirements, and OVAL Results Requirements sections below.

Basic Requirements

1.1 The language SHALL be capable of expressing the desired configuration state of a system.

1.2 The language SHALL be capable of expressing the actual configuration state of a system.

1.3 The language SHALL be capable of expressing where the actual system configuration differs from the desired configuration.

1.4 The language SHALL provide the ability to digitally sign any document.

OVAL Definition Requirements

The OVAL Definition Requirements serve to further refine Basic Requirement 1.1.

General Requirements

2.1 The language SHALL allow definitions to extend other definitions.

2.2 The various definition components SHALL be reusable.

2.3 The language SHALL contain the structure and the means to build unbounded logical combinations of individual components.

2.4 The language SHALL provide the ability to negate the logical statements.

2.5 The language SHALL provide the ability to represent specific configuration values with variables which can remain undefined until run-time.

Definition Document Requirements

2.6 The definition document SHALL contain the schema version against which it validates.

2.7 The definition document SHALL contain a timestamp for when it was created.

2.8 The definition document SHOULD contain information about the product name and version used to generate the definition file.

2.9 Multiple definitions SHALL be able to be contained in a single document.

2.10 The definition document SHALL contain all of the individual components used by each definition in the document.

Definition Content Requirements

2.11 Every OVAL Definition SHALL have a globally unique identifier.

2.12 All reusable components of a definition SHALL have a globally unique identifier.

2.13 Every globally unique identifier SHALL be structured to allow individual organizations to dynamically create identifiers without relying on an outside source and be ensured that global uniqueness is maintained.

2.14 The definition SHOULD contain at least one reference to an authoritative source for more information associated with the given issue.

2.15 An OVAL Test SHOULD be capable of testing all of the configuration parameters retrieved from a corresponding system element.

2.16 The labels used within an OVAL Test SHOULD mirror, in name and structure, those used on the system for a corresponding element.

OVAL System Characteristics Requirements

The OVAL System Characteristics Requirements serve to further refine Basic Requirement 1.2.

General Requirements

3.1 The system configuration items represented in an OVAL System Characteristics document SHOULD map directly to an object of concern within an OVAL Definition document.

System Characteristics Document Requirements

3.2 The system characteristics document SHALL contain the schema version against which it validates.

3.3 The system characteristics document SHALL contain a timestamp for when it was created.

3.4 The system characteristics document SHOULD contain information about the product name and version used to generate the definition file.

3.5 The system characteristics document SHALL provide information that uniquely identifies the specific system from which data is being collected.

System Characteristics Content Requirements

3.6 The System Characteristics document SHALL provide information about whether a specific item exists or does not exist on a system.

3.7 The System Characteristics document SHALL uniquely map items collected from a system to the definition objects to which they apply.

OVAL Results Requirements

The OVAL Results Requirements serve to further refine Basic Requirement 1.3.

General Requirements

4.1 The OVAL Results document SHOULD be capable of supporting different levels of detail in the results reported.

Results Document Requirements

4.2 The results document SHALL contain the schema version against which it validates.

4.3 The results document SHALL contain a timestamp for when it was created.

4.4 The results document SHOULD contain information about the product name and version used to generate the definition file.

4.5 The results document SHALL contain information that uniquely identifies the specific system being reported on.

Results Content Requirements

4.6 The result document SHALL contain the analysis result for each definition being reported upon in the document.

4.7 In the case where individual test information is being reported, the result document SHALL contain the analysis result for each test being reported upon in the document.

Additional Information

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