# Security Automation Developer Days – Summer 2011

## Table of Contents

# Introduction

The Security Automation Developer Days - Summer 2011 conference was held at MITRE in Bedford, MA. There were four presentations related to OVAL. The slides for these presentations are available on the OVAL website at: http://oval.mitre.org/community/developer_days.html. The notes below cover the discussions that occurred during and after the presentations, but do not provide thorough coverage of the presentations themselves. Please refer to the slides to gain a better understanding of the material that was presented.

## Note

One of the OVAL related discussions was led by CyberESI (OVAL for Artifact Hunting). CyberESI plans to provide minutes from their session, and therefore this document will cover only three of the four OVAL related discussions.

# OVAL Specification Review Minutes

## Why the OVAL Language Needs a Specification?

The OVAL Language needs a specification to clarify the ambiguities in the language. These ambiguities are caused by missing or conflicting documentation, specification by example, and overloaded terminology. Together these ambiguities make it difficult for those looking to learn the OVAL Language, write content, or develop tools that utilize the OVAL Language.

The primary goal of the OVAL Language specification is to make the language more accessible and easier to understand for members of the community. This involves separating the OVAL Language from its XML Schema representation which will remove any barriers caused by not knowing XML schema and allow for implementations in other representations (e.g. JSON, etc.). Furthermore, it will eventually serve as the authoritative documentation for the OVAL Language. It will also help to update and consolidate information about the OVAL Language which was previously distributed across the OVAL Language Schemas, OVAL Forum Archives, and OVAL Web Site.

## Revised OVAL Language Use Cases and Requirements

As part of the development of the OVAL Language specification, the OVAL Language use cases[1] were revised. The primary change made to the use cases was the addition of use case scenarios which provide specific examples of how the OVAL Language can be used. The revised OVAL Language use cases were posted to the oval-developer-list for review on 4/29/2011[2]. The OVAL Language requirements were also revised to be more binding agnostic. Specifically, the references to OVAL Content being a document were removed. The updated requirements were posted to the oval-developer-list on 5/9/2011 for review[3].

## OVAL Data Model Overview

The OVAL Language Data Model defines what OVAL Constructs and OVAL Enumerations are and what they are composed of. The OVAL Language Data Model contains models for the OVAL Language Core Schemas (oval-common-schema, oval-definitions-schema, oval-system-characteristics-schema, oval-

results-schema, and oval-directives-schema). This distinction represents a shift away from the OVAL Language being all of the OVAL Language Core and Component Schemas. With this approach, the OVAL Language Component Schemas are simply extensions of the OVAL Language Core schemas. In the OVAL Language Data Model, constructs will be defined using textual descriptions, UML diagrams, and a table of properties and enumerations will be defined using textual descriptions and a table with the valid values in the enumeration.

## What If Something is Deprecated?

The OVAL Language contains constructs and enumeration values that have been deprecated. The OVAL Language Deprecation Policy[4] states:

"When an OVAL Language construct is marked as deprecated its usage becomes strongly discouraged and it may be removed in a later release."

In our current draft of the OVAL Language Data Model, we decided not to include the deprecated constructs or enumerations to make the specification cleaner. However, it is important to note that even if a construct is deprecated, it can be used until it is finally removed from the OVAL Language. The community was asked for thoughts on this decision.

The community stated that it would be beneficial to include the OVAL Language Deprecation Policy and the OVAL Language Versioning Methodology in the specification as appendices. Furthermore, the community mentioned that constructs should only be removed from the specification when they are removed from the OVAL Language. This is because the specification is needed to make people aware of the deprecated constructs, why they were deprecated, and what constructs, if any, can be used in place of the deprecated construct. Lastly, it was suggested that the OVAL Language Deprecation Policy definition should be modified such that it states "…constructs *will* be removed in a later release." rather than "…constructs *may* be removed in a later release.". By making this change, it will explicitly indicate that the deprecated constructs will be removed and that they should not be relied upon.

## How to Handle Constructs Defined Outside of OVAL?

In the OVAL Language Requirements, OVAL Language Requirement 3.1.4.1 states that the language must provide a mechanism to ensure the integrity and authenticity of all content written in the language. The current XML Schema representation of the OVAL Language uses XML Signatures to fulfill this requirement. The community was asked if it made sense to include signature properties in the current constructs that utilize XML Signatures in the OVAL Language Data Model even though the specification was geared towards being binding agnostic.

One member of the community stated that if it is supported by the OVAL Language Schemas then it should be documented in the specification. It was also pointed out that the implementation of the OVAL Language must provide a mechanism to ensure the integrity and authenticity of the content, but, tools are not required to implement it.

The OVAL Language specification represents a big shift from the XML Schema representation of the OVAL Language to a more logical binding agnostic view of the OVAL Language.

One member of the community recommended that since XML Digital Signatures exist in a separate namespace they should be allowed anywhere in a document such that they can be used or not used as needed. This would make it such that the allowed locations for XML Digital Signatures would not need to be defined in the specification.

Another member of the community stated that the goal of a specification is to provide standardization, a common interpretation, and interoperability and if we do not define how specific elements work we will not achieve those goals. They also stated that we need to document the XML Schema binding because that is what we have and if we are going to get interoperability we need to understand what these things mean. That means if we are going to include external constructs in the specification we will need to reference the normative guidance for the constructs and we should not over specify the constructs on top of the normative guidance.

It was also pointed out that this is not a matter of referencing or not referencing, but, rather are the external constructs captured in the logical model of the specification or in the binding section of the specification.

It was also recommended that the use of XML Digital Signatures be deprecated as well as explicit metadata and instead just allow people to use metadata where they see fit rather than trying to predict what metadata will be useful. Over the years, additional metadata is added because you have chosen to limit the information that resides in other namespaces which would keep it separate from the OVAL Language and tools could just use this information if they wanted to.

Another member of the community disagreed with deprecating XML Digital Signatures be by defining it in specific locations tools will need to or at least should support it and know where to look rather than relying on some out of band negotiation. This is the opposite direction of where we want to go with similar things in this domain.

Since there were differing opinions on this topic, this discussion has been continued over the oval-developer-list and can found at the following link.

http://making-security-measurable.1364806.n2.nabble.com/OVAL-Specification-topics-td6521019.html

## How to Handle Container Constructs?

The OVAL Language contains many container constructs that are an artifact of the current XML Schema representation. An example of this is that an OVAL Definitions Document, in current XML Schema representation, can contain zero or one DefinitionsType construct(s). The DefinitionsType construct can then contain one or more DefinitionType constructs. The community was asked if it made sense to include the DefinitionsType container construct or simply specify that OVAL Definitions Document can contain zero or more DefinitionType constructs.

The overwhelming response from the community was that the OVAL Language specification should align closely with the XML Schema representation of the language and should include the container constructs. The primary reasons that were given for taking this approach were that the community is

primarily concerned with the current XML Schema representation of the language and by abstracting it out to a more logical view it may result in a specification that is harder to understand as well as cause synchronization issues where you have to make sure everything is consistent between the logical view and the XML Schema representation which is challenging. Lastly, some members of the community expressed that, from experience, trying to separate the logical view from its biding representation was challenging and really ended up resulting in twice as much work. It was also recommended that the specification should initially focus on the XML Schema representation of the language to minimize any difficulties when transitioning from the OVAL Language Schemas to the OVAL Language specification.

It was also asked if there was a driver behind separating the logical representation of the OVAL Language from its current XML Schema representation. The primary driver here was to make the OVAL Language more accessible for those who may not have a background in XML Schema as well as open the door for future innovation such as using a different binding representation.

## How to Handle the Datatypes?

The OVAL Language datatypes are currently defined by the DataTypeEnumeration which is a union of the SimpleDatatypeEnumeration and the ComplexDatatypeEnumeration. The SimpleDatatypeEnumeration contains all simple data which can be represented as a string value whereas the ComplexDatatypeEnumeration contains structured data such as the record datatype. In the OVAL Language specification, we decided not to maintain this separation of the datatypes into distinct groups (e.g. simple vs. complex), but rather just represent all datatypes in the DataTypeEnumeration because they are semantically the same thing. The community was asked if this decision made sense and there was consensus that this representation of datatypes was appropriate for the OVAL Language specification.

## What to do with OVAL Language Metadata Constructs?

The ElementMapType and DeprecatedInfoType constructs from the oval-common-schema are defined to provide structured metadata about the OVAL Language. Specifically, the ElementMapType construct describes the relationship between an OVAL Test, OVAL Object, OVAL State, and OVAL Item and the DeprecatedInfoType construct describes why a construct was deprecated and in what version of the OVAL Language. The community was asked whether or not these constructs belonged in the OVAL Language.

One member of the community stated that since the metadata constructs are defined in oval-common-schema, which is part of the OVAL Language, they should be included somewhere in the specification.

Another member of the community considered metadata constructs as constructs of convenience in terms of describing the structure of a document using a particular schema language and are not part of the OVAL Language and should not be included as part of the specification unless they are included by reference.

A member of the community disagreed that they are simply a construct of convenience because they help schema authors and tool developers who may process or create OVAL content. As a result, the specification should provide clear guidance on how to use these constructs.

Another member of the community mentioned that it might be useful to simply define these constructs in the appendix that contains the OVAL Language Deprecation policy.

A member of the community also mentioned that these constructs add no value to an OVAL instance document and should not be included.

Lastly, another member of the community disagreed that they provide no value because an XML Schema document is an instance document and these constructs are used to describe it. If an XML Schema is considered as an instance document having these constructs defined apply to use cases such as in the case of tools.

Due to the differing opinions regarding this topic, this discussion has been continued over the oval-developer-list and can be found at the following link

http://making-security-measurable.1364806.n2.nabble.com/OVAL-Specification-topics-td6521019.html

## How to Represent xsd:choice Constructs?

The current XML Schema representation of the OVAL Language utilizes the xsd:choice construct to allow for one element, from a collection of elements, to be present in an XML instance document. An example of when the xsd:choice construct is used is in the ComponentGroup construct which can be an object_component, variable_component, literal_component, or any one of the OVAL Functions. In a UML diagram, the ComponentGroup could be represented as a composition of zero or more of these constructs or it could be represented as an inheritance relationship where each construct is derived from the ComponentGroup construct. In the OVAL Language specification, they are currently represented as a composition relationship in the specification. The community was asked if there was a preference in how they are represented. A member of the community stated that treating the xsd:choice construct as a composition relationship seems to align with the XML Schema representation.

## Other Discussion Topics

Due to time constraints, the OVAL Language Specification Review session ended after the discussion of the “How to Represent xsd:choice Constructs?” topic. The discussion, for the remaining topics, has been continued over the oval-developer-list and can be found at the following link.

http://making-security-measurable.1364806.n2.nabble.com/OVAL-Specification-topics-td6521019.html

## References

[1] OVAL Language Use Cases
http://oval.mitre.org/adoption/usecasesguide.html

[2] OVAL Language Use Cases Discussion
http://making-security-measurable.1364806.n2.nabble.com/FOR-REVIEW-Revised-OVAL-Language-Use-Cases-td6316500.html

[3] OVAL Language Requirements Discussion
http://making-security-measurable.1364806.n2.nabble.com/FOR-REVIEW-Revised-OVAL-Language-Requirements-td6345171.html

[4] OVAL Language Deprecation Policy
http://oval.mitre.org/language/about/deprecation.html

# PowerShell Proposal

This session was led by Kelly Hengesteg, Jeffrey Snover, and Michael Tan from Microsoft.

## Introduction

By Kelly Hengesteg

Microsoft began writing OVAL content for Exchange and SQL Server and realized PowerShell was the only means to examine some of the configuration items for those products. It was also determined that more products from Microsoft will use PowerShell as the basis for configuration in the future. However, OVAL does not support PowerShell so a proposal to extend OVAL to support PowerShell was devised.

The presentation is intended to provide an overview to PowerShell, and to introduce the proposed design for integrating PowerShell into OVAL.

## PowerShell Overview

By Jeffrey Snover

Historically, UNIX had a very powerful shell and Microsoft had a very poor shell. The PowerShell team's intent was to create a powerful new shell incorporating interaction with common Windows constructs – WMI, registry, et al.

With the Common Engineering Criteria (http://www.microsoft.com/cec/en/us/cec-overview.aspx), Microsoft is specifying PowerShell as the mechanism for the configuration management of their products and systems moving forward. PowerShell Cmdlets will support configuration, operation verification tests, lifecycle, diagnostics, and data management for their products and systems.

PowerShell has been embraced by the virtual machine community, e.g. VMWare, because the number of machines people need to manage has exploded with the wide adoption of virtualization.

At its core, PowerShell has an automation engine DLL that gets hosted and accessed in various ways – interactive shell, hosted inside programs like Exchange. The PowerShell engine operates on either a string (which is parsed) or a known data structure. In the OVAL proposal, a schema that invokes the automation engine as an API is used.

Jeffrey then presented an overview of the execution architecture of PowerShell comparing it with UNIX and VMS DCL. He showed a few demonstrations of PowerShell commands.

He also explained some of the restrictions that can be placed on the execution environment of Cmdlets. A "constrained runspace" allows detailed configuration of what can and cannot be executed.

## Discussion

Question: On what versions of Windows is PowerShell available?
Answer: PowerShell 2.0 is available on Windows 7 & 2008 R2. XP and above can run PowerShell 2.0.

Question: In a constrained runspace, is it possible to discover the available commands?
Answer: Yes.

Question: How do Cmdlets map to the messy storage mechanisms at the lower layers?
Answer: Execution units supply a contract to PowerShell engine and what they do, PowerShell does not care.

## Proposal

By Michael Tan

***NOTICE: Unfortunately, the microphone was off during this portion of the presentation undermining the development of detailed minutes. Please email oval@mitre.org if you have any corrections or additions to this section.***

Michael led a detailed review of the PowerShell cmdlet_test proposal that he sent to the oval-developer-list (http://making-security-measurable.1364806.n2.nabble.com/Windows-PowerShell-Proposal-for-SCAP-tp6464505p6464505.html).

## Discussion

***NOTICE: There were several questions, but again, the microphone was off and the answers cannot be heard in the recording.***

Question: Is piping of commands supported, or can I use more than one select?
Answer: We are not modeling a full PowerShell pipeline. Generally, you will only need the Gets (get-<*noun*> cmdlets). We allow you to use the Select to identify the fields in a response from a Get, but we do not support a full pipeline. If that capability is needed we can enhance the proposal to support the capability. However, doing so increases the risk of malicious use of PowerShell and we do not think the capability is needed at this time.

Question: Is there anything that will prevent the assessment of the cmdlet_test from a remote host?
Answer: No, PowerShell cmdlets can be remotely executed.

Question: Will the cmdlet_object support variables on its entities, like other OVAL Objects do?
Answer: Yes, the entities in the cmdlet_object proposal are just like all other entities in the OVAL Language. As standard OVAL entities the var_ref attribute will be supported.

Question: Could OVAL be used to, at run time, get the module name and supply that value as a variable to the module name entity?
Answer: Yes, there is a PowerShell cmdlet, Get-Module, that can be used to get the module information. This information could then be used to as a value for a variable on another cmdlet_object?

Question: Is it possible to inject commands via abuse of the ';' character or other characters?
Answer: One of the goals in developing PowerShell was to specifically avoid such a possibility. There is only one place that this can be done, and it is called Invoke-Expression. This cmdlet has been reoved from the allowed set of cmdlets.

Question: Is the OVAL cmdlet_object proposal intended to provide input to the PowerShell DLL via a data structure or a string?
Answer: The proposal is essentially populating a data structure that is supplied to the DLL. However, tools can really take either approach to interfacing with PowerShell.

Remark: Notice that this is the first occurrence of the record data structure in an OVAL Object data structure. This structure is currently used in OVAL States only.

Remark: The list of allowed cmdlet verbs has been specifically whitelisted to include only those that do not impact the system state. We filtered out all cmdlets that are known to have side effects. We have limited the allowed verbs via OVAL Language Schema restrictions.

Remark: The Microsoft proposal includes sample code that demonstrates the usage of the proposed cmdlet_object.

Question: Can a cmdlet be signed?
Answer: Yes, cmdlets can be signed to allow them to run in a restricted runspace and you can check the signatures of cmdlets before execution.

Question: What is the import verb?
Answer: import is a way of getting data from one state and pulling it into a system. This verb will allow you to read from various files.

Question: A goal of OVAL has been to look at the primary source where data is stored not secondary interfaces that mirror the data. In PowerShell, is there anything that might lead to differing results based upon how the data was retrieved?
Answer: Yes, that is possible with any access layer. If you don't get your response from the single source of truth then there is a possibility of caching causing problems if you query other interfaces. However, the fact is that the underlying configuration data is inconsistently stored. PowerShell cmdlets are intended to provide a single interface to this configuration data.
Remark: There are several known configuration items that can only be checked via PowerShell cmdlets because the data is held in a proprietary data store.

Remark: As Windows evolves there will be new cmdlets added, but it is unlikely that existing administration capabilities will be removed. This means that there will still be registry access, and API

access to password policy information. We will simply be adding in more new capability with PowerShell cmdlets, not replacing capability.

Question: Will Microsoft produced baselines use the cmdlet_test exclusively?
Answer: No, the plan is to continue using legacy OVAL Tests whenever possible and only use the new cmdlet_test when a legacy OVAL Test is not available. If the community wants a 100% PowerShell baseline, that is possible, but we would need to hear from the community that it is needed or preferred.

Question: Will Microsoft's security tool support the export of baselines using cmdlet_objects in a hybrid or exclusive mode?
Answer: For now, the tool will export hybrid content. If there is significant demand we could consider exporting cmdlet only content but some settings are not available through cmdlets and there is significant vendor investment in the legacy OVAL Objects so we feel we should continue to support those legacy objects.

Remark: PowerShell does not replace WMI. WMI is a standards based approach to management. It is not being replaced. WMI and PowerShell are complementary.

Question: What does a query look like when no select statement is present?
Answer: As a best practice, the select statement in the cmdlet_object should always be used to specify which fields should be represented in the OVAL Item. Without a selection, various engines might represent the data differently.
Remark: Outputting consistent results is key.

Question: Will the OVAL Interpreter prototype code be made available?
Answer: Yes, as soon as it is completed we will post the prototype on the OVALDI sourceforge.net project. There is also a code sample in the Microsoft proposal.

Remark: We would like to see this proposal folded into version 5.10.

Question: Can you show an example of getting a low level configuration setting?
Answer: To clarify, the applications that run on Windows are much further along in the adoption of PowerShell than Windows itself. Therefore, for most Windows configuration items the legacy API is likely to be the only interface to the configuration data.
Remark: the PowerShell proposal includes examples and references to complete listings of cmdlets for several applications.

## SQL Test Overview

The ind-def:sql57_test and ind-def:sql_test have been in existence for some time, but have not been widely implemented. This discussion provided some real world experiences and challenges with the test, specifically the ind-def:sql57_test.

Matt Hansbury began by providing an overview for the IRS/SCAP content development, including history, plans, and challenges as background for the discussion. Rob Hollis, from ThreatGuard, followed

up with an in depth conversation regarding the ind-def:sql57_test's unique challenges and lessons learned. He presented a series of challenges, designed to spark discussion and re-evaluation of some aspects of the test for possible future versions of OVAL.

## Database Applicability

Once of the challenges faced in using the ind-def:sql57_test was determining which database(s) a given OVAL Test applied to. While talking about how to determine which checks to run on specific databases (such as Oracle, MSSQL, etc.), Rob pointed out that ThreatGuard used the affected element at the OVAL Definition level to provide this hint. This effectively uses the metadata in the definition to determine applicability. Additionally, it was mentioned that the applicability capabilities that have been discussed within the OVAL Community of late would also be a potential solution here (see discussion on "applicability_check" http://making-security-measurable.1364806.n2.nabble.com/Proposal-for-extending-Oval-criteria-criterion-and-extend-definition-to-specify-applicabilityChecks-tp6271556p6271556.html).

## Database Connection String

Another point of discussion was the connection string. Proper use of the connection string has never really been fully fleshed out. There are long standing concerns about the open ended nature of the entity leading content developers to expose database credentials. However, there is a need for great flexibility in defining connection parameters because the access level and how a tool connects to a database can greatly affect the assessment results. One suggestion was to parameterize the connection string entity to allow more flexible connectivity. While this could be useful in some cases, it wasn't applicable to the IRS case, since the target connection is made before the OVAL Definition is encountered. This experience leads to another option which be to simply define the connection information separately from the OVAL Object.

Additionally, it was asked if tasking could help with the connection string issue. Tasking is seen as an emerging area that will be aggressed by the enterprise reporting group as they work on specifications like ARF and AI. However, it was pointed out that if you add too much of this to the tasking, then you lose the opportunity to tailor things on a per OVAL Definition basis and you introduce a dependency upon tasking in OVAL.

## Other Topics

It also came up that there were use cases where the content author would need to be able to specify not only the target system to connect to, but also the specific database instance.

The idea that we might need separate tests for Oracle, MSSQL, etc. was also raised. This would allow for variations of SQL since nearly all major databases have their own extension of SQL.

Another commenter mentioned the distinction between assessing the configuration settings of the database server, vs. assessing an application's settings that happen to reside in a database instance. This served to highlight the fact that there were a number of use cases under consideration for this test.

During the discussion it was pointed out that anonymous database field names should not be used. For example, SELECT COUNT(*), should be replaced with SELECT COUNT(*) AS total. This is explicitly documented in the schema documentation for the ind-def:sql_test. If fields are not named there will be inconsistencies across database server implementations. This worked for the set of databases that have been tested, but it is not clear if it will work across all database implementations. However, a follow up point was made that we currently write most OVAL Definitions to specific platforms, so this wasn't viewed as a major issue. We simply need to acknowledge which platforms we are targeting when writing a given test.

Later, while discussing the discovery issue, it was mentioned that in many cases, authors need more control over which database instances are assessed. Sometimes one needs to assess all instances of a database, in other cases, a more granular approach is required. Tasking was brought up as a possible solution here, as well. It was observed that the optimal solution here may ultimately involve a hybrid solution involving both formal and informal constructs to accommodate the appropriate use cases.

While discussing the challenges of assessing multiple databases within a database server, it was pointed out that here is no documented convention for handling the result of each instance assessment. This is an area that requires more community discussion. One solution that was offered was to leverage the OVAL System Characteristics system_info element to distinguish instances. The system_info element is intended to identify the asset that was assessed. Perhaps we need to consider the asset being assessed to be the specific database instance instead of the operating system which was running the database server. This could be implemented today leveraging the AI specification in the xsd:any space of the system_info element. This would allow tools to create a single OVAL Results document that represented the full assessment of each database instance.

Finally, the question was asked by Matt Hansbury if we could agree to make the connection_string entity optional. It was pointed out that to do so would make it such that the information regarding the connection string would be entirely removed from OVAL. Therefore it would be better to keep the connection string information as mandatory for now.

## Conclusion

At the end of the discussion is became clear that while it was possible to implement the ind-def:sql57_test in a real world environment, it was necessary to make a number of assumptions in order to properly execute the checks.

The community needs to continue to gather use cases for the tests to try to fully understand all of the required components and options that is required of the test. Once this information is truly understood, one or more revisions should be made to the test, in order for it to better serve all of the community's needs.