The MITRE Corporation

IT Security Automation Conference – OVAL Future Considerations Workshop

September 29, 2010

Workshop Summary

Welcome

The attendees were welcomed to the OVAL Workshop at the 2010 IT Security Automation Conference.

Background

The background portion of the workshop began by looking at the different ways that we can ensure long term success for the OVAL Language. The first way that we can do this is to limit the barriers to adoption making it easier for organizations, even those outside the world of SCAP, in the vulnerability management, security advisory, and configuration management space to use the OVAL Language in their products and services. Additionally, we can ensure success for the OVAL Language by focusing on our core competencies and making sure that we are able to solve the problems of the OVAL Community. Also, in this workshop, we would like to look at where we are today, where we want to go as we move forward, and consider if a major version is necessary and practical. Lastly, as always, in this workshop we are looking to hear back from you to get your input on these discussion topics, how to improve the OVAL Language, and hear your concerns as we move forward.

The OVAL Community has acquired a lot of experience since it began back in 2002. From its early days as SQL schemas to its current implementation as XML schemas, the OVAL Language has continued to progress across major revisions with its latest being Version 5.0 in 2006. Since then, we have had on-going discussions regarding a new major revision, Version 6.0, and its implications. However, as Version 5.x has been in use longer and longer, organizations have invested more into this particular major revision. As a result, if we want to move to a new major revision of the OVAL Language, we need to make sure that we have a very strong business case for doing so.

In preparation for this workshop, we have realized that the core schemas of the OVAL Language (oval-definitions-schema, oval-system-characteristics-schema, and the oval-results schema) have not changed much since the release of Version 5.0. This tells us that the OVAL Language has been quite stable over the past few years. Over time, many organizations in the OVAL Community have implemented the capabilities outlined in the OVAL Adoption Program and there are many organizations looking to purchase tools that use the OVAL Language. Lastly, there are mandates, policies, and programs requiring the use of OVAL.

Next, the OVAL Language Version Methodology was reviewed. Under the Version Methodology, any change to the OVAL Language requires that the OVAL Language version be incremented. If the change to the OVAL Language invalidates existing OVAL content, it constitutes a major version. If the change to the OVAL Language does not invalidate existing OVAL content, it constitutes a minor version. However, there is an exception to the rule that invalidating existing content requires a major version change. The exception states that critical defects can be fixed in a minor version of the OVAL Language. This rule is in place to address bugs in the schema that result in nonsensical or invalid content. The Version Methodology also states that deprecated constructs, after they have been deprecated for at least one version, can be removed from the OVAL Language in either a major or minor version as defined in the

OVAL Deprecation Policy. It is also important to note that moving to a major version of the OVAL Language does not mean that it is necessary to make changes that will force vendors to completely rewrite their products.

Discussion

To begin the discussion portion of the workshop, the attendees were asked to consider the following questions.

- Are there changes that would facilitate greater adoption?
- How do we scale more efficiently?
- Can OVAL be made more easily extensible?
- What would make OVAL more maintainable?

Next, a high-level overview of the following possible major revision changes was presented.

- Define one test type.
- Consolidate Object and State entities.
- Leverage a common asset model.
- Remove datatype definitions from the common schema
- Integrate CPE

Then the topic of defining a single test type in the OVAL Language was discussed in greater detail. Currently there is an OVAL Test defined for every test (currently 136 tests as of Version 5.8) in the OVAL Language.   Each OVAL Test associates an OVAL Object with zero or more OVAL States and allows the number of OVAL Items that need to exist on the system and the number of collected OVAL Items that need to match the OVAL State(s) to be defined (using the check_existence and check attributes respectively). Since all of the OVAL Tests are the same outside of the name and documentation, these test constructs could be consolidated into a single test construct effectively reducing the size of the schema by about one-third. Additionally, existing OVAL Content could easily be translated to use this single test type with a stylesheet.

[Uncertain Speaker] In the context of removing all tests and replacing with one, where would the test-specific information reside?

[Jon Baker] OVAL has a notion of tests, objects, and states. You would have a single test element that would refer to a registry_object and a registry_state. All tests would be the same and the test-specific information would reside in the objects and states. Currently, the OVAL Interpreter does not distinguish between different types of tests and all tests are evaluated the same.

[Todd Dolinsky] We do this similarly to the OVAL Interpreter and I think that we are fine with it. We just need to be careful that we do not mix objects and states of different types.

[Todd Dolinsky] How would we enforce the object/state pairings if we only have a single test?

[Jon Baker] Schematron rules would still exist to enforce those relationships.

Next the topic of consolidating OVAL Object and OVAL State entities was discussed. This topic was initiated as a request, in Version 5.8, to allow for any OVAL State entity to appear in its respective OVAL Object. If this change was made, it would have represented a fundamental change to how objects work in that they would no longer represent the minimum required set of entities necessary to uniquely identify an OVAL Item on a system. Additionally, this change would introduce a source of duplication. To address this issue and streamline the collection of OVAL Items on a system for checks like "No world writable files are permitted", an unbounded filter element was added to all OVAL Objects in Version 5.8. While this provides a solution to the problem, it is still somewhat cumbersome. As a result, we should consider what changes we could make under a major revision. It would be possible to simplify and reduce the size of the OVAL Language by defining a single structure that could serve as both an OVAL Object or an OVAL State depending on the context in which it was used. If the structure was referenced as an OVAL Object, in an OVAL Test, it would specify the OVAL Items on a system to collect and if the structure was referenced as an OVAL State, in an OVAL Test, it would specify the expected state of the collected items.

The potential directions that the OVAL Language could take, as we move forward, were then discussed. The first direction presented was to continue with minor revisions of OVAL 5.x indefinitely. Under this direction, the OVAL Language would continue to grow adding features over time. However, it would do so at a much slower rate using the OVAL Deprecation Policy to remove capabilities as time progresses. The primary downside to this approach is that constructs must be deprecated for at least one revision before they can be removed and we could end up carrying around a fair amount of deprecated constructs for quite some time. The next direction that was presented was to commit to develop and maintain OVAL 5.x for some number of years while simultaneously developing OVAL 6.0. The main uncertainty here is whether or not the OVAL Community can in fact support two versions of OVAL over the oval-developer-list, in the OVAL content, in the OVAL Interpreter as well as in the other tools that support OVAL. The last direction that was discussed was to develop a new major release and archive Version 5.x of the OVAL Language which would allow for introduction of new capabilities without the need to deprecate unused constructs. The downside to this approach is that the current investment of the OVAL Community in the existing OVAL Content and capabilities may be lost depending on the actual changes that are introduced in the major revision. It would be possible to mitigate these potential losses by having a long release candidate and repository transition phase to provide additional time for content to be updated and issues with the capabilities to be worked out.

[Richard Whitehurst] It doesn't seem like you need a major change to make all objects like states and deprecate the old objects.

[Jon Baker] Yes, we could do that and we could continue to add/remove capabilities with the deprecation policy and continue through Version 5.x indefinitely.

[Jon Baker] What do you think when you hear a major revision?

[Todd Dolinsky] I don't cringe. I think the single test and having objects like states is useful. The hardest part is training new employees on OVAL. If the middle or bottom option makes content generation easier then I am all for it.

[Jon Baker] Are there other things that we should consider?

[Melissa Albanese] Ultimately, we are going to have to go to 6.0. Depending on the changes in 6.0, if there is a stylesheet to ease content transition from 5.x to 6.0, it would make things better.

[Richard Whitehurst] I like the first option where everyone merges to a more stable version and does not use the deprecated tests.

[Harold Booth] To be clear you will need the infrastructure to support 6.0.

[Jon Baker] We could do a long release candidate before archiving version 5.x.

[Harold Booth] SCAP will force you to support old content. We should have a plan for supporting both versions for some time. Specifically, we will need to maintain content in both versions for some time, but would not recommend actively revising both versions (no adding new tests to 5.x branch).

[Jon Baker] we could maintain the OVAL Interpreter on 5.x until SCAP moves off that version and maybe longer.

[Richard Whitehurst] We cannot tell their customers to upgrade to a 6.x tool. We will have to support both for some time.

Workshop Action Items

The following action items were then announced to the workshop attendees.

- Develop a proposed direction for OVAL releases.
- Develop an OVAL Language specification.
- Continue to discuss the proposed questions over the oval-developer-list.
- Post minutes for this workshop.

Workshop Conclusion

The attendees were encouraged to get involved in the advancement and development of the OVAL Language by joining the mailing lists or participating in the OVAL Adoption program if they had a product or service that used the OVAL Language. It was also announced that the minutes for this workshop will be released within the next few days. The attendees were thanked for attending the workshop.