HS SEDI
Homeland Security Systems Engineering and
Development Institute

OPEN VULNERABILITY AND ASSESSMENT LANGUAGE
OVAL®

Future Directions

Jon Baker
September 29, 2010

Homeland
Security

Workshop Focus

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 is succeeding, how do we ensure long term success?**
  - Limit barriers to adoption
  - Maintain focus on our core competencies (avoid distractions)
  - Ensure that OVAL continues to solve the community's problems

- **Let's explore the need for a major revision**
  - Consider where the OVAL Language is today.
  - Consider possible options for continued advancement of OVAL.
  - Overview of some possible capabilities.

- **Looking for community feedback**
  - help us all understand the current strengths and weakness
  - let us know about your concerns

A Bit of History

- **The OVAL Community has years of experience**
  - Started work on OVAL in December 2002
  - Version 3 - July 2004
  - Version 4 - April 2005
  - Version 4.1 - August 2005
  - Version 4.2 - December 2005
  - Version 5.0 - June 16, 2006

- **Major revisions have been discussed since the release of version 5.0**
  - Both strong supporters and critics of another major revision.
  - There must be a strong business case for a major release.

Current State of Version 5.x

- **Fairly stable with only minor capabilities and tests added with each new release.**
- **Broad community adoption**
  - Many products have implemented OVAL Capabilities
  - Organizations look to buy tools that use OVAL
  - Policies and mandates requiring OVAL 5.x
  - Numerous programs relying upon OVAL
- **Many repositories of OVAL Definitions**
  - Large investments in content and tools to create content
  - Public repositories and private repositories
  - Organizations have invested in OVAL Content

Challenge: Balance opportunities for improvement with investment in capabilities and content.

What constitutes a major or minor version?

- **Any modification to the OVAL Language requires the version to be incremented.**

- **Major vs. Minor – Content is the key.**
  - **Minor Version:** Modifications that do not invalidate OVAL content
  - **Major Version:** Modifications that invalidate existing OVAL content

- **For every rule, there is an exception.**
  - **Deprecation Policy:** deprecated constructs may be removed
  - **Critical Defects:** critical defects may be corrected

- **A major revision does not have to be that major.**

Questions

Are there changes that would facilitate greater adoption?

- Need to understand the implementation challenges with OVAL today.
- Make OVAL easier to implement.

How do we scale more efficiently?

- Need to create component schemas for many more platforms.
- Schema development is community driven.
- No single team will be an expert in all platforms.

Can OVAL be made more easily extensible?

- The core schemas have been very stable.
- The component schemas change often.
- Component schema additions and revisions require OVAL Language revisions.
- Well documented with example extensions including defined criteria for a valid extension
  - Could the development of extensions the be federated?

What would make OVAL more maintainable?

- All tests are essentially identical.
- The meaning of <trustee_sid/> is defined 23 times in OVAL 5.8.
- At a minimum the meaning of a given object entity is defined three times.

Directions to Consider

- **Continue with minor revisions to OVAL 5.x indefinitely.**
  - Fold in refinements and capabilities as needed utilizing the deprecation policy to remove capabilities over time.
  - Slow and steady progress will be made.
  - Will carry a lot of baggage.

- **Commit to developing and maintaining OVAL 5.x for some number of years while developing OVAL 6.**
  - Can the community really support two versions of OVAL?
    - developer list activity (new tests, component schemas, etc for two versions)
    - OVAL content, OVAL Interpreter, authoring tools all supporting two versions

- **Develop a new major release and archive version 5.x.**
  - Investment in existing content may be lost.
  - Investment in existing capabilities may be lost.
  - Consider a long release candidate phase and delayed repository transition.

Major Revision Possibilities

There are significant opportunities to refactor to improve maintainability, extensibility, and reduce vendor implementation burden.

- **Define only one test type**
- **Consolidate Object and State Entities**
- **Leverage a common asset model**
- **Remove datatype definitions from the common schema**
  - Datatypes don’t always apply across platforms
  - Adding a new datatype forces a revision to the common schema
- **Integrate CPE**
  - Integration of Affected Platforms and Criteria
  - CPE for affected platforms and products
  - Generator use CPE

One Test Type

- 136 Tests Defined in Version 5.8.
- Tests associate an object with a state(s).
- Define how many items must satisfy the object and how many of those matching items must satisfy the state(s).

```
<xsd:element name="version_test" substitutionGroup="oval-def:test">
  <xsd:annotation>
      <xsd:documentation>...</xsd:documentation>
        <xsd:appinfo>
              <oval:element_mapping>...</oval:element_mapping>
        </xsd:appinfo>
        <xsd:appinfo>
            <sch:pattern id="catos-def_version_test">
                  <sch:rule context="catos-def:version_test/catos-def:object">
                        <sch:assert test=“...“>the object child element must reference a version_object</sch:assert>
                  </sch:rule>
                  <sch:rule context="catos-def:version_test/catos-def:state">
                        <sch:assert test=“...“>the state child element must reference a version_state</sch:assert>
                  </sch:rule>
            </sch:pattern>
      </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexType>
      <xsd:complexContent>
          <xsd:extension base="oval-def:TestType">
              <xsd:sequence>
                  <xsd:element name="object" type="oval-def:ObjectRefType" />
                  <xsd:element name="state" type="oval-def:StateRefType" minOccurs="0" maxOccurs="unbounded"/>
              </xsd:sequence>
          </xsd:extension>
      </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Consolidate Object and State Entities

- **Version 5.8 Request: Allow any OVAL State entity to appear in an OVAL Object.**
  - A fundamental change in how objects work.
  - All object entities would become optional.
  - Introduces another source of duplication.

- **Result: Added ability to filter objects with states in Version 5.8**
  - Streamlines data collection for checks like, “No world writable files are permitted.”
  - Still a bit cumbersome.

- **Major version suggestion: Define one structure where its use is context sensitive.**
  - When referenced by a test as an object this new structure would specify the set of items to collect on a system.
  - When referenced by a test as a state this new structure would specify the expected state of any collected items.
  - Tremendous reduction of schema and simplification of the language.

What Next?

- **Develop a proposed direction for OVAL releases.**
  - Socialize with the OVAL Board and the community.

- **Develop an OVAL Language specification.**
  - Better describe what we have today.
  - Provide insight into opportunities for refinement.

- **Work with the community to understand strengths, weaknesses, and impact.**

- **Post discussion minutes.**

https://oval.mitre.org/oval/about/developer_days.html

Are there changes that would facilitate greater adoption?

- Need to understand the implementation challenges with OVAL today.
- Make OVAL easier to implement.

How do we scale more efficiently?

- Need to create component schemas for many more platforms.
- Schema development is community driven.
- No single team will be an expert in all platforms.

Can OVAL be made more easily extensible?

- The core schemas have been very stable.
- The component schemas change often.
- Component schema additions and revisions require OVAL Language revisions.
- Well documented with example extensions including defined criteria for a valid extension
  - Could the development of extensions the be federated?

What would make OVAL more maintainable?

- All tests are essentially identical.
- The meaning of <trustee_sid/> is defined 23 times in OVAL 5.8.
- At a minimum the meaning of a given object entity is defined three times.

Get Involved!

- **Join the OVAL mailing lists**
  - **OVAL-Announce** – General news and announcements about OVAL.
  - **OVAL Developer's Forum** – Public forum for discussing the OVAL Language, addressing OVAL implementation issues, and for assisting other developers with OVAL.
  - **OVAL Repository Forum** – Public forum for discussing OVAL Repository content.

  https://oval.mitre.org/community/registration.html

- **Participate in the OVAL Adoption Program**
  - Help shape the effort

  https://oval.mitre.org/adoption/