OVAL – language for experts

Matěj Týč, Marek Haičman
Security Compliance @ Red Hat

Contents

- How is it to get started w. OVAL.
- What are the first impressions of OVAL newbie.
- OVAL and true security.

Where to learn about OVAL?

- oval.mitre.org: Allegedly “archived”, contains lang reference
- ovalproject.github.io: Has tutorial, but no reference docs.
- oval.cisecurity.org: Not a learning resource

- Github CISecurity
- Github OVAL-Community
- Github OVALProject

Bad signal-to-noise ratio

- The Tutorial example (only packages from the whitelist are allowed) ~ 60 lines without comments.

Bad signal-to-noise ratio


```
– <oval_definitions xsi:schemaLocation=" http://oval.mitre.org/XMLSchema/oval-definitions-5 oval-definitions-schema.xsd http://o
  http://oval.mitre.org/XMLSchema/oval-common-5 oval-common-schema.xsd http://oval.mitre.org/XMLSchema/oval-definitions-5#un
  definitions-schema.xsd">
  – <generator>
      <oval:product_name>Tutorial Example Generator</oval:product_name>
      <oval:schema_version>5.11</oval:schema_version>
      <oval:timestamp>2014-12-21T04:42:18.845-05:00</oval:timestamp>
    </generator>
  – <definitions>
    – <definition id="oval:tutorial:def:1" version="1" class="compliance">
      – <metadata>
          <title>RPM WhiteList</title>
          <description>Fail if anything not on the whitelist is installed</description>
        </metadata>
      – <criteria>
          <criterion comment="Test to check that only listed packages are installed." test_ref="oval:tutorial:tst:1"/>
        </criteria>
      </definition>
    </definitions>
  – <tests>
    – <rpminfo_test id="oval:tutorial:tst:1" version="1" check="all" check_existence="none_exist" comment="all packages">
        <object object_ref="oval:tutorial:obj:2"/>
      </rpminfo_test>
    </tests>
  – <objects>
    – <rpminfo_object id="oval:tutorial:obj:2" version="1" comment="Filtered Packages">
        <name datatype="string" operation="pattern match">.*</name>
        <oval-def:filter action="exclude">oval:tutorial:ste:1</oval-def:filter>
      </rpminfo_object>
    </objects>
  – <states>
    – <rpminfo_state id="oval:tutorial:ste:1" version="1">
        <name datatype="string" operation="equals" var_ref="oval:tutorial:var:1"/>
      </rpminfo_state>
    </states>
  – <variables>
    – <constant_variable id="oval:tutorial:var:1" version="1" datatype="string" comment="Package Names">
        <value>termcap</value>
        <value>auditd</value>
        <value>libselinux</value>
      </constant_variable>
    </variables>
  </oval_definitions>
```

Bad signal-to-noise ratio

- The Tutorial example (only packages from the whitelist are allowed) ~ 60 lines without comments.
- Too much for a Hello World example!
- Equivalent Python code: 10 lines of self-explanatory code.

Difficult concepts

- Variables are lists.
- Tests have check and check_existence. Check is required, but it is not used if there is no state in the test.

Difficult concepts

- Variables are lists.
- Tests have check and check_existence. Check is required, but it is not used if there is no state in the test.
- criteria use the extended logic (pass, fail, notchecked, …) with AND / OR / NOT operators.
- OVAL can reference other variables, tests and so on – without a supportive tooling, it is hard to have everything at one place.

Language vision

- OVAL is more fine-grained.
  - Python: assert **a** == **b**
  - OVAL: There is a **test** that **object a** conforms to **state b** using the equality relation.
- But not always:
  - OVAL: textfilecontent54 object determines files to be examined AND regular expression.
  - Python: Files and a function evaluating them would be separate entities.

Language vision

- C: Functions return error code, take pointers as arguments.
- C++: Compatible with C, but you have objects and a really wide range of possibilities how to do things.
- Python: import this # shows the Zen of Python
- OVAL: declarative language

Limitations – config. compliance

- Simple OVAL is OK to check for “default installation” errors or admin omissions.
- Configuration means config files, but textfilecontent test/object/state combinations are not powerful enough
  - Config file ordering issues
  - Multi-line comments
  - Include statements

New tests

For proper configuration compliance, much more special tests are needed. What about this:

1) SW authors expose config-parsing libraries.
2) Security Compliance SMEs formulate the object/state interfaces.
3) The developed test is shared with the community.
4) What about the OVAL sandbox?

Supportive tooling

- Provide test environment with scenarios.
- Extract rule’s OVAL to one place.
- Visualize OVAL results.
- Debug OVAL evaluation.