OVAL for Artifact Hunting

Charles Schmidt

July (Friday) 13, 2012

What is OVAL for Artifact Hunting?

- Describe and test assertions about system state related to malware presence
  - No change to OVAL intent; new use case for that intent
- Provide a way to turn malware analysis into open, actionable, objective, multi-platform, and vendor-neutral testing instructions
  - Open – clearly document test criteria for all to review/understand
  - Actionable – directly usable for testing in enterprise
  - Objective – tests are unambiguous and deterministic
  - Multi-platform – do not need to learn new languages/structures when switching platform
  - Vendor-neutral – there are existing proprietary techniques for this; want to be able to do this without being bound to a vendor

What this effort is not

- No plan to change existing use cases (compliance, vulnerability, inventory scanning)
  - There may be cross-pollination of capabilities, but we expect some distinction between these use cases in practice
- No plan to include artifact hunting definitions in existing OVAL Repository
  - A repository of artifact hunting definitions (if one was created) would have different audience and management procedure
- Not an attempt to replace signature-based malware scanners
  - There is a role for both capabilities meeting different needs
- No plan to expand OVAL into network-based scanning
  - Artifact hunting will focus on host-based indicators

Definition Generation

- One significant goal is automatic generation of OVAL tests from standards such a MÆC and OpenIOC
  - Already have some demonstrations of this
- Facilitate automatic test generation from malware analysis descriptions
  - Standards-based malware analysis becomes instantly actionable in an enterprise with no added burden to analysts
- Manual generation also must be supported
  - Not the goal to bind malware standards to OVAL
  - Automatic generation just an obvious value-add we are pursuing

Why OVAL?

- Testing state assertions is OVAL's goal – this is what we are trying to do here
  - OVAL already provides many of the desired features
  - OVAL meets our criteria: open, actionable, objective, multi-platform, and vendor-neutral
- No reason to create yet-another-language
  - Force users to learn
  - Force implementers to support
  - Force someone to pay for development and maintenance
- Good chance for synergy with other use cases
  - Artifact hunting already well served by tests developed for configuration/vulnerability testing
  - Some tests developed for artifact hunting may prove useful to other OVAL use cases

Example

- **Excerpt from a file using existing OVAL capabilities to detect Zeus indicators**

```
<win-def:file_object version="1" id="oval:maec_out:obj:10">
    <win-def:path>C:\Documents and Settings\Administrator\Local Settings\Temporary
Internet Files\Content.IE5\WDUF49AN</win-def:path>
    <win-def:filename>truegate[1].htm</win-def:filename>
</win-def:file_object>
<win-def:file_object version="1" id="oval:maec_out:obj:11">
    <win-def:path>C:\Documents and Settings\Administrator\Local Settings\Temporary
Internet Files\Content.IE5\WDUF49AN</win-def:path>
    <win-def:filename>webhp[1].htm</win-def:filename>
</win-def:file_object>
<win-def:registry_object version="1" id="oval:maec_out:obj:12">
    <win-def:hive>HKEY_USERS</win-def:hive>
    <win-def:key>S-1-5-21-842925246-1425521274-308236825-
500\SOFTWARE\Microsoft\Benee</win-def:key>
    <win-def:name xsi:nil="true"/>
</win-def:registry_object>
```

Example use

- Preceding OVAL definitions are not positive identifiers
  - Just identifies system artifacts associated with Zeus
  - Was auto-generated from MÆC – does not distinguish between relative value of indicators
    - Further manual refinement of OVAL could be more expressive
- Currently material still useful
  - Help filtering out false-positives
  - Help convey information about malware artifacts
- Use would increase with the ability to make assertions about a wider variety of system artifacts
- Just using OVAL for its intended purpose – identify presence of specific system artifacts

Activities

- Presentation last year at Developer Days by CyberESI
  - Outlined ways to extend OVAL to meet artifact hunting use case – Windows Portable Executables, File System, etc.
  - Talk generated significant discussion – both interest and concerns
- OVAL Sandbox currently has a section for the development of artifact-hunting tests
  - Allow development without contaminating main OVAL Language
  - Only a handful of new tests developed thus far
- Initial analysis of OVAL coverage relative to other standards

Moving Forward

- Identification and prioritization of OVAL capability gaps
  - Currently, primary source will be malware description standards
    - Standards represent community consensus on malware information
  - Other input welcome
- Develop tests
  - Schema, interpreter, and conversion scripts (for content generation)
- Evaluate utility
  - Is there an improvement over current practice? For whom?
- Keep in sandbox until community decides otherwise
  - Afterwards, may still be reason to segregate from support of other use cases

Specific Near-Term Targets

- Expand Windows File States
  - Digital signature info, strings from binary, NTFS data streams, other file timestamps
- Process58 States
  - Open handles, strings in process memory
- New tests
  - Windows Mutexes
  - Windows Task Scheduler
  - Windows Prefetch Entries
  - Windows Drivers
  - Windows Event Log Entries
  - Windows Routing Table Entries
  - Windows System Restore Items
  - Windows Executable File Object (adding imports, exports, and sections)

Summary

- Plan is to move forward with targeted experimentation
  - Stay within sandbox for now
  - Regular evaluation of utility of any changes
- Primary focus will be on solidifying links between OVAL and malware standards (MÆC, OpenIOC, etc.)
  - Better meet the goal of automated test generation
- Periodic checks back in with the community
  - Consider how to integrate into official OVAL, if deemed useful

Questions?