Draft 1.0
06/16/2016

Open Vulnerability and Assessment Language

Brett Gurman, Hardik Jhaveri, Rujuta Palande

A. Introduction

Is Cyber Security a myth? This question keeps on swinging both sides now and then. Everyday new efforts are being developed to create a secure environment for users to leverage the advances of technology without any compromise. OVAL which stands for Open Vulnerability and Assessment Language is another example of a practice created by **MITRE** and supported by the information security community towards a secure cyberspace. OVAL helps to standardize how to assess computer systems and report their status.

B. Why use OVAL?

Determining the existence of software vulnerabilities, configuration issues, programs, and patches in local systems had no structured means because of which system administrators and other end users faced problems. They were restricted to analyzing text-based descriptions which was a labor-intensive and error-prone process to determine if any issue existed on the local system.

OVAL solves these problems. The widespread availability of OVAL will promote standardized vulnerability and configuration assessment and will provide consistent and reproducible information assurance metrics.

OVAL helps you to determine which vulnerabilities or configuration issues exist on your system. You may then use this information to obtain appropriate software patches and fix information for remediation. Hence, OVAL acts as a preventive measure. As an addition, you can use a vulnerability database which includes information about vulnerabilities that OVAL does not, such as the severity of the problem, whether it is locally or remotely exploitable, remediation information, and so on.

C. Components of OVAL

1. OVAL Language

OVAL uses Extensible Markup Language (XML) because of its data centric approach which makes it easier to extract data and is machine readable. The main purpose of the OVAL Language is to standardize the three key steps of the assessment process,

- Representing configuration information of systems for testing
- Analyzing the system for the presence of the specified machine state
- Reporting the results of this assessment

The OVAL community has developed three schemas approved by the OVAL Board written in Extensible Markup Language (XML) to serve as the framework and vocabulary of the OVAL Language. An OVAL System Characteristics schema represents system information; an OVAL Definition schema expresses an ideal specific machine state, and an OVAL Results schema reports the results of the assessment performed.

OVAL Definitions are machine readable files written in XML to highlight any system vulnerability, configuration issue, programs and patches present on the system. For these assessments we have different definition files which give us a roadmap.

- OVAL Vulnerability Definitions — Tests that determine the presence of vulnerabilities on systems.
- OVAL Compliance Definitions — Tests that determine whether the configuration settings of a system meets a security policy.
- OVAL Inventory Definitions — Tests that whether a specific piece of software is installed on the system.
- OVAL Patch Definitions — Tests that determine whether a particular patch is appropriate for a system.
- OVAL Miscellaneous Definitions – Tests that do not fall into any of the four main classes.

OVAL System Characteristics are generated by the OVAL Interpreter. The Core System Characteristics Schema defines all operating system independent objects. It contains general information about the system that data was collected from, including information that can be used to identify the system. All the objects that have been collected for assessment and their status are mentioned in this. Finally after the system and object is mapped, the current state of that object is specified for analyses.

OVAL Results schema stores the results of the evaluations performed on the system against the gold standard definitions. The current state of the machine is compared to the OVAL Definitions to check for vulnerabilities or issues. The data generated can be interpreted to take the necessary actions to mitigate the vulnerabilities and configuration issues such as alter system configuration settings, install patches, or take external precautions to limit access to the affected systems. OVAL Results file can be of two types - full and thin. A value of 'thin' means only a minimal amount of information is provided about the assessment of objects whereas a 'full' means that very detailed information is provided allowing in-depth reports to be generated from the results. The interpreter also generates Results to HTML file which converts an OVAL Results document into a more readable html format.

2. OVAL Repository

The OVAL Repository is the central meeting place for the OVAL Community to discuss, analyze, store, and disseminate OVAL definitions. The OVAL Language and any resulting OVAL content based upon the language that is stored in the OVAL Repository are free to use by any organization or individual for any research, development, and/or commercial purposes. The OVAL Repository Moderator evaluates and reviews definitions for publication in the OVAL Repository. Once new definitions are published in the OVAL Repository they are subject to community review. The OVAL Repository uses the publicly known vulnerabilities identified in the Common Vulnerabilities and Exposures which describe a list of vulnerabilities and exposures. Hence, the OVAL Repository plays a very important role in maintaining OVAL as an international, information security, community effort.

3. OVAL Interpreter

The OVAL Interpreter is hosted on SouceForge.net, is a freely available reference implementation that demonstrates the evaluation of OVAL Definitions. Based on a set of XML

Definitions the Interpreter collects system information, evaluates it, and generates a detailed OVAL Results file. When we run the Interpreter, it will provide a list of OVAL Definition IDs and their results on the system.

D. Working

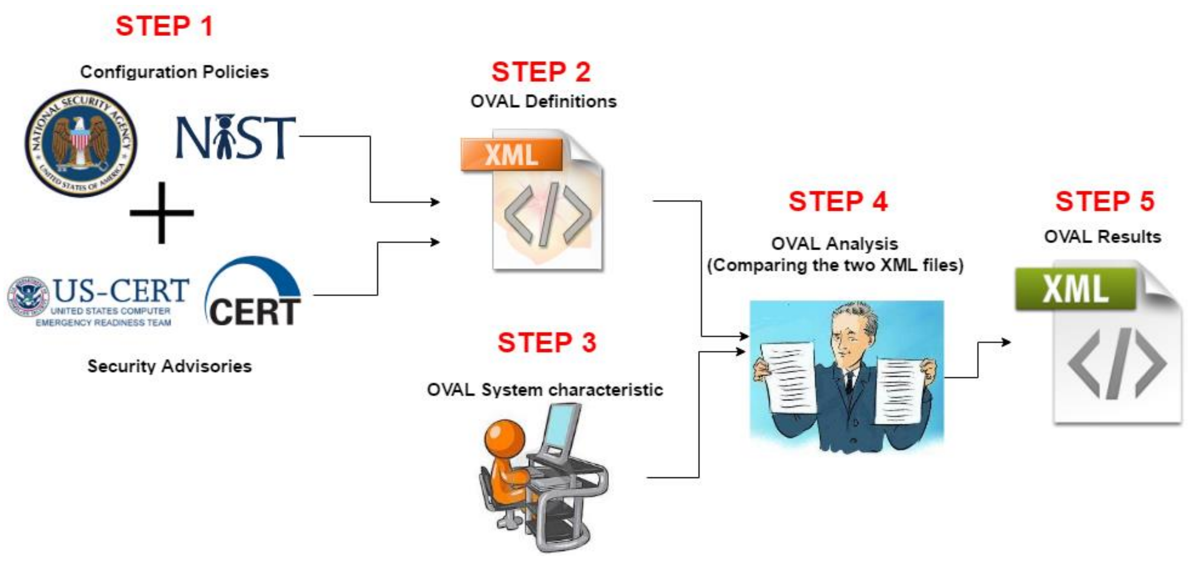


Figure 1: Working of OVAL

Step 1

a. Configuration policies: Certain government agencies such as NSA (national security agency) and NIST (National institute of standards and technologies) develop "Best Practices" policy. Following these policies ensures system security.
b. Security advisories: CERT-CC (computer emergency response team – co-ordination center), US-CERT (United States – computer emergency readiness team), and other organizations publish security advisories. These advisories warn of current threats and system vulnerabilities.

These best practices and advisories are used to create an OVAL definition file for a particular system.

Step 2

Specific machine configuration details from the security advisories and configuration policy documents are extracted and encoded as an OVAL Definition. The OVAL Definition schema is used to define the XML framework for writing:

a. OVAL Vulnerability Definitions by defining the conditions that must exist on a computer for a specific vulnerability to be present
b. OVAL Patch Definitions by defining the conditions on a computer that determine whether a particular patch is appropriate for a system
c. OVAL Compliance Definitions by defining the conditions on a computer that determine compliance with a specific policy or configuration statement.
d. OVAL Inventory Definitions by defining the conditions on a computer that determine whether a specific piece of software is installed on the system.

Step 3

This file basically encodes the details of the system to be evaluated. The OVAL System Characteristics schema defines a standard XML format for representing system configuration information, which includes operating system parameters, installed software application settings, and other security relevant configuration values. The schema provides a list of system characteristics against which OVAL definitions can be compared in order to analyze a system for the presence of a particular machine state. By utilizing the provided OVAL System Characteristics file, other applications would not need to perform data collection, but rather can use the provided information to perform analysis.

Step 4

The OVAL Definitions from *Step 2* and the System Characteristics from *Step 3* are compared to determine if the current system state is vulnerable or not vulnerable, by using an OVAL interpreter.

Step 5

As mentioned in the Step 4, based on the set of definitions, the interpreter collects system information, evaluates it, and generates detailed OVAL Results file. Results of analysis are formatted as an OVAL Results document.

E. Case Study

The purpose of this section is to provide real-world examples of what the OVAL files should look like. Examples of the definition file, system characteristic file, and results file will be shown.

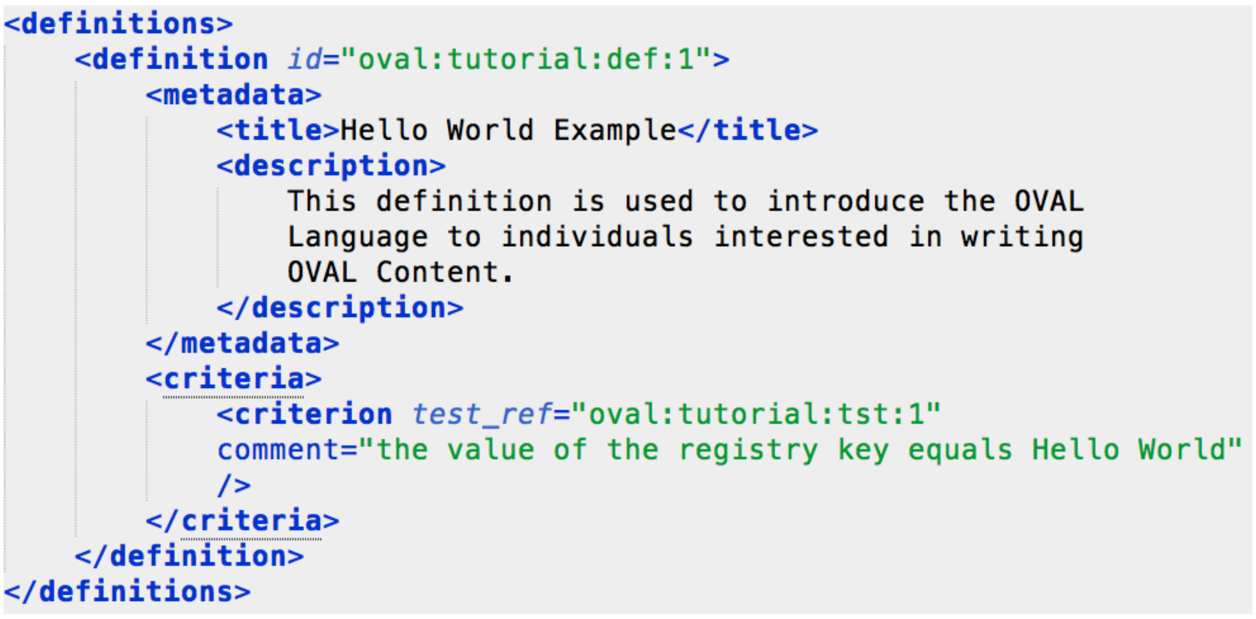
1. Definition File

1.1 Hello World Example

The “Hello World” example will give the reader a sense of intuition for the structure of OVAL.

- Definitions

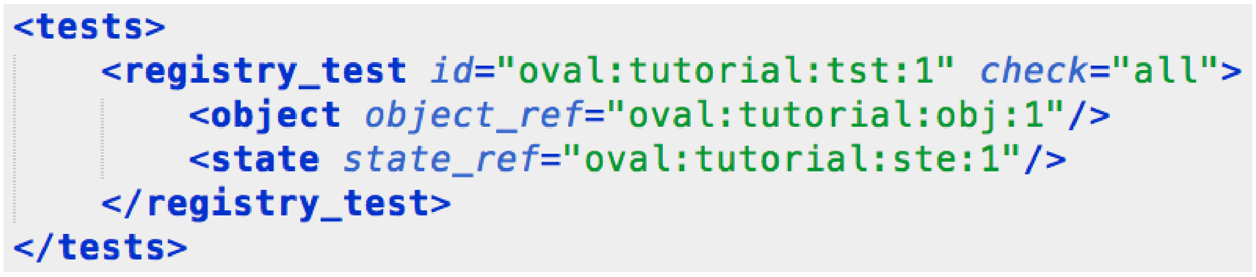
At the core of the definitions file is a list of definitions, each consisting of metadata and criteria. The metadata is where the title and description of the definition are specified. The criteria are a list of one or more criterion, as well as an operator to combine them if there are multiple. Each criterion refers to an individual test, along with an optional comment.



```
<definitions>
    <definition id="oval:tutorial:def:1">
        <metadata>
            <title>Hello World Example</title>
            <description>
                This definition is used to introduce the OVAL
                Language to individuals interested in writing
                OVAL Content.
            </description>
        </metadata>
        <criteria>
            <criterion test_ref="oval:tutorial:tst:1"
            comment="the value of the registry key equals Hello World"
            />
        </criteria>
    </definition>
</definitions>
```

- Tests

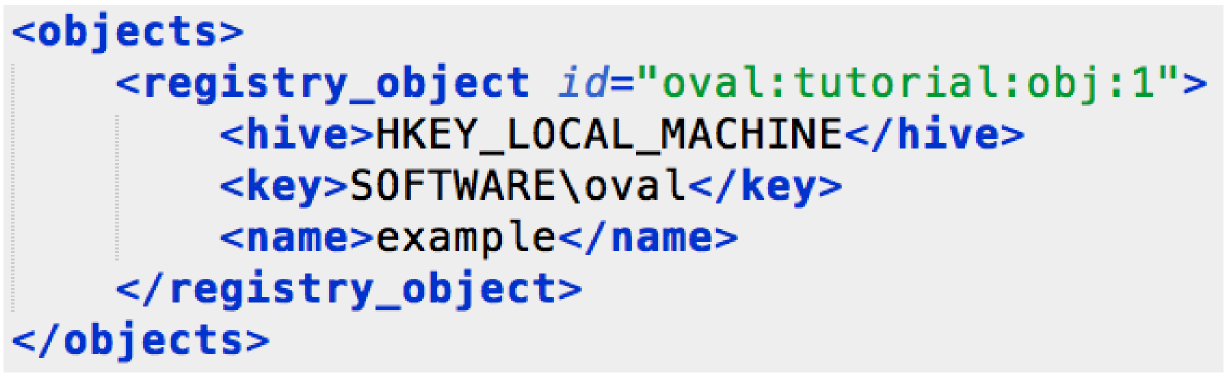
The tests section of an OVAL definitions file contains a list of one or more tests that are used to check the state of specified objects. Each test has an ID, and contains a reference to an object and a reference to a state to be checked.



```
<tests>
    <registry_test id="oval:tutorial:tst:1" check="all">
        <object object_ref="oval:tutorial:obj:1"/>
        <state state_ref="oval:tutorial:ste:1"/>
    </registry_test>
</tests>
```

- Objects

The objects section of an OVAL definitions file contains a list of one or more objects, or actual things on a system that are to be checked. Each entry contains an ID, as well as a hive, key, and name, which are all used to locate the object in the system.



```
<objects>
    <registry_object id="oval:tutorial:obj:1">
        <hive>HKEY_LOCAL_MACHINE</hive>
        <key>SOFTWARE\oval</key>
        <name>example</name>
    </registry_object>
</objects>
```

- States

The states section of an OVAL definitions file contains a list of one or more states, each of which describes the state an object must have for test to be considered TRUE. Each entry has an id, and contains a value, as well as optionally a datatype field.

```
<states>
    <registry_state id="oval:tutorial:ste:1">
        <value>Hello World</value>
    </registry_state>
</states>
```

1.2 Real example

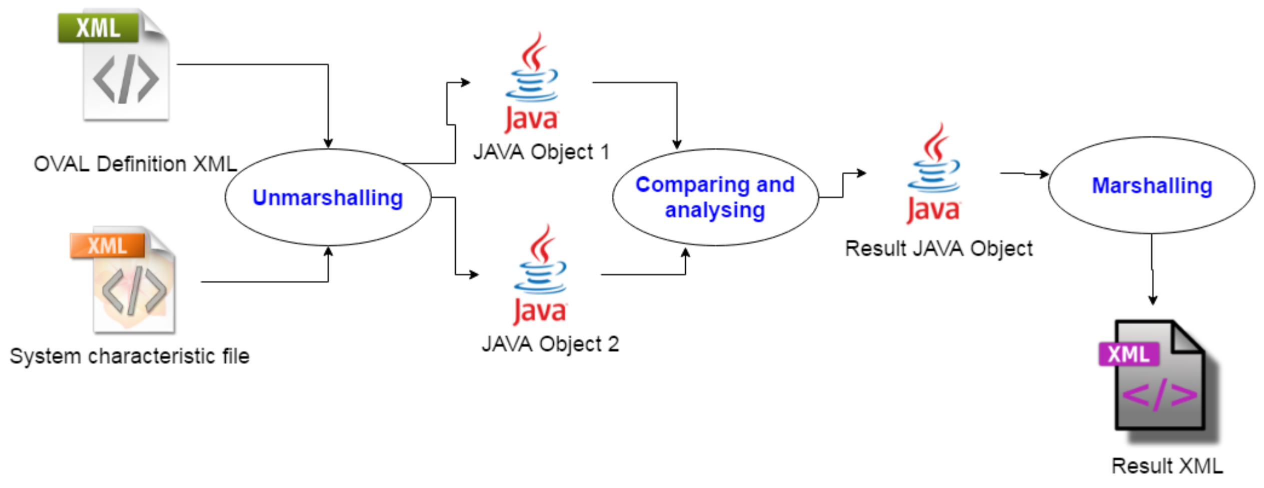
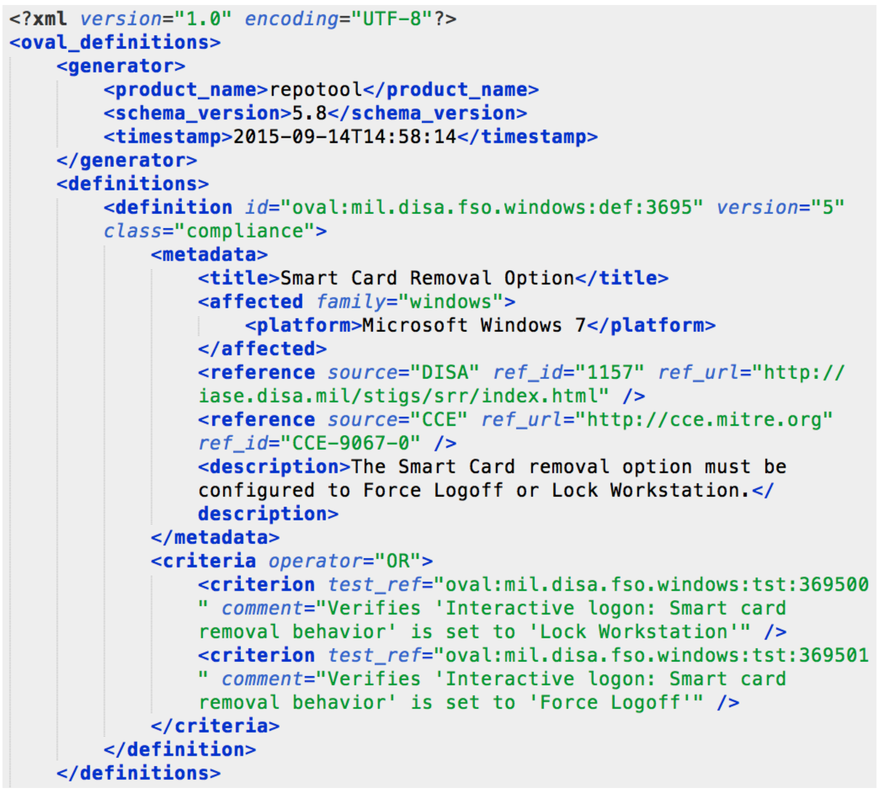
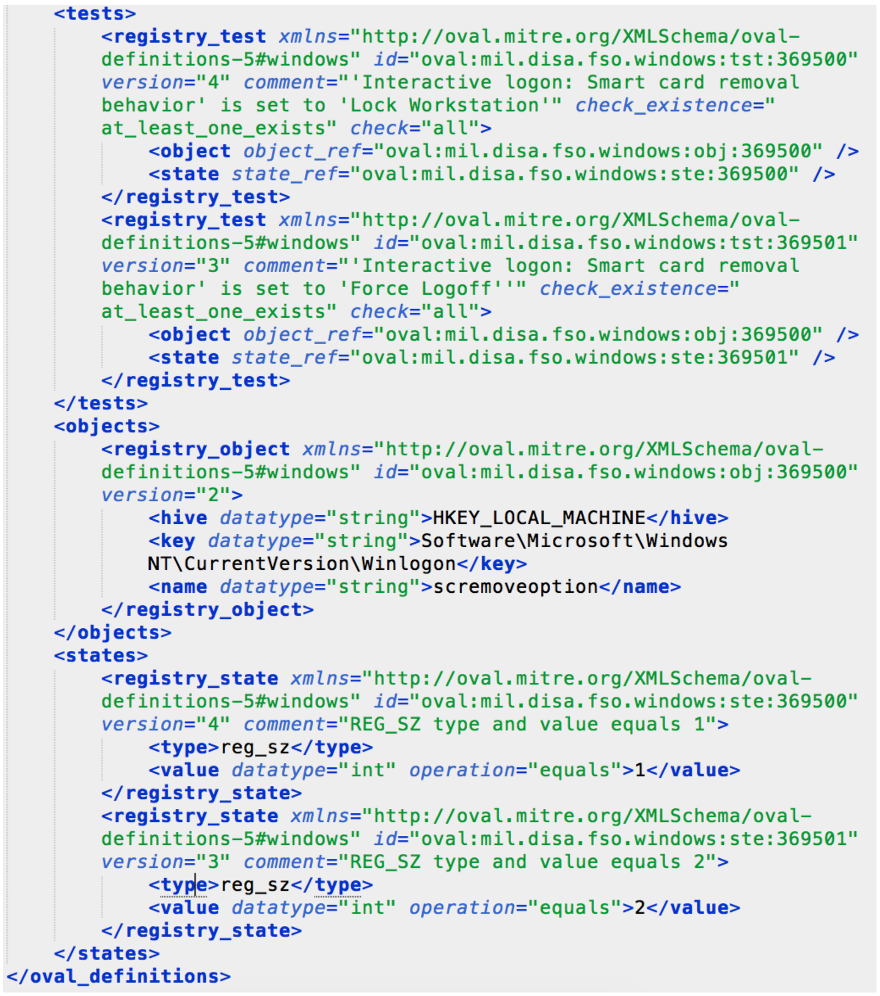


Figure 2: Steps to obtain the result xml

This file is very similar to the Hello World example in the previous section, with a few minor differences. The only section not included in the Hello World example is the Generator section. The purpose of this section is simply to provide some information about how the OVAL document was created in the definitions section, the definition includes some additional metadata that further describes the definition. It also contains some references, which are optional as well. Under the criteria tag, we see that we now have multiple criterions - when this is the case, an operator is provided in the criteria tag that indicates how to join the results of the individual tests. Registry states are expressed in a detailed way. In this example, they contain an additional type tag, which further specifies how to interpret the value.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<oval_definitions>
    <generator>
        <product_name>repotool</product_name>
        <schema_version>5.8</schema_version>
        <timestamp>2015-09-14T14:58:14</timestamp>
    </generator>
    <definitions>
        <definition id="oval:mil.disa.fso.windows:def:3695" version="5"
        class="compliance">
            <metadata>
                <title>Smart Card Removal Option</title>
                <affected family="windows">
                    <platform>Microsoft Windows 7</platform>
                </affected>
                <reference source="DISA" ref_id="1157" ref_url="http://
                iase.disa.mil/stigs/srr/index.html" />
                <reference source="CCE" ref_url="http://cce.mitre.org"
                ref_id="CCE-9067-0" />
                <description>The Smart Card removal option must be
                configured to Force Logoff or Lock Workstation.</
                description>
            </metadata>
            <criteria operator="OR">
                <criterion test_ref="oval:mil.disa.fso.windows:tst:369500
                " comment="Verifies 'Interactive logon: Smart card
                removal behavior' is set to 'Lock Workstation'" />
                <criterion test_ref="oval:mil.disa.fso.windows:tst:369501
                " comment="Verifies 'Interactive logon: Smart card
                removal behavior' is set to 'Force Logoff'" />
            </criteria>
        </definition>
    </definitions>
```



```xml
    <tests>
        <registry_test xmlns="http://oval.mitre.org/XMLSchema/oval-
        definitions-5#windows" id="oval:mil.disa.fso.windows:tst:369500"
        version="4" comment="'Interactive logon: Smart card removal
        behavior' is set to 'Lock Workstation'" check_existence="
        at_least_one_exists" check="all">
            <object object_ref="oval:mil.disa.fso.windows:obj:369500" />
            <state state_ref="oval:mil.disa.fso.windows:ste:369500" />
        </registry_test>
        <registry_test xmlns="http://oval.mitre.org/XMLSchema/oval-
        definitions-5#windows" id="oval:mil.disa.fso.windows:tst:369501"
        version="3" comment="'Interactive logon: Smart card removal
        behavior' is set to 'Force Logoff'" check_existence="
        at_least_one_exists" check="all">
            <object object_ref="oval:mil.disa.fso.windows:obj:369500" />
            <state state_ref="oval:mil.disa.fso.windows:ste:369501" />
        </registry_test>
    </tests>
    <objects>
        <registry_object xmlns="http://oval.mitre.org/XMLSchema/oval-
        definitions-5#windows" id="oval:mil.disa.fso.windows:obj:369500"
        version="2">
            <hive datatype="string">HKEY_LOCAL_MACHINE</hive>
            <key datatype="string">Software\Microsoft\Windows
            NT\CurrentVersion\Winlogon</key>
            <name datatype="string">scremoveoption</name>
        </registry_object>
    </objects>
    <states>
        <registry_state xmlns="http://oval.mitre.org/XMLSchema/oval-
        definitions-5#windows" id="oval:mil.disa.fso.windows:ste:369500"
        version="4" comment="REG_SZ type and value equals 1">
            <type>reg_sz</type>
            <value datatype="int" operation="equals">1</value>
        </registry_state>
        <registry_state xmlns="http://oval.mitre.org/XMLSchema/oval-
        definitions-5#windows" id="oval:mil.disa.fso.windows:ste:369501"
        version="3" comment="REG_SZ type and value equals 2">
            <type>reg_sz</type>
            <value datatype="int" operation="equals">2</value>
        </registry_state>
    </states>
</oval_definitions>
```

2. System Characteristic file

The system characteristic file contains information about the actual state of a system. Collecting the values needed for this file involves manually locating each value on the system.

- Collected Objects

The collected objects section contains a mapping between the objects listed and the results file and the actual values listed in the next section of the System Characteristic file, System data. Each entry consists of a flag, which indicates whether or not the author had success collecting the necessary value, an object id, which specifies an object from the definitions file, and an item reference, which corresponds to a single entry in the system data section.

- System Data

The system data section contains the actual values for the relevant objects from the definitions file. Each entry contains an ID, which maps to an entry in the collected objects section (and further to an object from the definition file), and a value, possibly with data-type specified.

```
<system_data>
    <file_item id="1" xmlns="http://oval.mitre.org/XMLSchema/oval-system-
    characteristics-5#windows">
        <value datatype="int" operation="equals">0</value>
    </file_item>
</system_data>
```

3. Results file

The schema for the results file is very simple. The only section is the results section, which consists of a definitions section containing a list of definitions, each with an ID, identifying it as one of the definitions from the definitions file, and a result, indicating true or false. The definitions section will contain one entry per definition in the definitions file.

```
<results>
    <definitions>
        <definition id="oval:mil.disa.fso.windows:def:3695" version="5" result="
        false"/>
    </definitions>
</results>
```

F. Conclusion

Hence, we can say that Open Vulnerability and Assessment Language is truly 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.