Security Automation and Continuous Monitoring
Internet-Draft
Intended status: Informational
Expires: March 11, 2017

M. Cokus
D. Haynes
D. Rothenberg
The MITRE Corporation
J. Gonzalez
Department of Homeland Security

September 7, 2016

OVAL(R) Results Model
draft-cokus-sacm-oval-results-model-01

Abstract

This document specifies Version 5.11.1 of the OVAL Results Model which is used to express the results of an evaluation of a set of systems based on a set of OVAL Definitions and the target systems' OVAL System Characteristics.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 11, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Open Vulnerability and Assessment Language (OVAL) [OVAL-WEBSITE] is an international, information security community effort to standardize how to assess and report upon the machine state of systems. For over ten years, OVAL has been developed in collaboration with any and all interested parties to promote open and publicly available security content and to standardize the representation of this information across the entire spectrum of security tools and services.

OVAL provides an established framework for making assertions about a system's state by standardizing the three main steps of the assessment process: representing the current machine state; analyzing

the system for the presence of the specified machine state; and representing the results of the assessment which facilitates collaboration and information sharing among the information security community and interoperability among tools.

This draft is the part of the OVAL contribution to the IETF SACM WG that standardizes the representation of the results of an evaluation. It is intended to serve as a starting point for the endpoint posture assessment data modeling needs of SACM specifically Evaluation Results.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. DirectivesType

The DirectivesType defines what result information has been included, and to what level of detail, in the OVAL Results, for each possible result value defined in the ResultEnumeration.

Property	Type	Count	Description
definition_true	DirectiveType	1	Defines what result information has been included for OVAL Definitions that evaluate to 'true'.
definition_false	DirectiveType	1	Defines what result information has been included for OVAL Definitions that evaluate to 'false'.
definition_unknown	DirectiveType	1	Defines what result

			information has been included for OVAL Definitions that evaluate to 'unknown'.
definition_error	DirectiveType	1	Defines what result information has been included for OVAL Definitions that evaluate to 'error'.
definition_not_evaluated	DirectiveType	1	Defines what result information has been included for OVAL Definitions that evaluate to 'not evaluated'.
definition_not_applicable	DirectiveType	1	Defines what result information has been included for OVAL Definitions that evaluate to 'not applicable'.

Table 1: DirectivesType Construct

3. DefaultDirectivesType

The DefaultDirectivesType defines the result information to include in the OVAL Results for all OVAL Definitions regardless of class as defined in the ClassEnumeration.

Property	Type	Count	Description
include_source_definitions	boolean	0..1	Specifies whether or not the source OVAL Definitions are included in the OVAL Results. When 'true' the source OVAL Definitions MUST be included in the OVAL Results. When 'false' the source OVAL Definitions MUST NOT be included in the OVAL Results. Default Value: 'true'

Table 2: DefaultDirectivesType Construct

4. ClassDirectivesType

The ClassDirectivesType defines the result information to include in the OVAL Results for a specific class of OVAL Definitions as defined in the ClassEnumeration. Please note that this will override the directives in the DefaultDirectivesType for the specified class.

Property	Type	Count	Description
class	oval:ClassEnumeration	1	Specifies the class of OVAL Definitions to which the defined OVAL Results directives will be applied.

Table 3: ClassDirectivesType Construct

5. DirectiveType

The DirectiveType defines what result information, and to what level of detail, is included in OVAL Results.

Property	Type	Count	Description
reported	boolean	1	Specifies whether or not OVAL Definitions, with the specified result, should be included in the OVAL Results. If the reported property is set to 'true', OVAL Definitions that evaluate to the specified result MUST be included in the OVAL Results. If the reported property is set to 'false', OVAL Definitions that evaluate to the specified result MUST NOT be included in the OVAL Results.
content	ContentEnumeration	0..1	Specifies the level of detail that is included in the OVAL Results. Default Value: 'full'

Table 4: DirectiveType Construct

6. ResultsType

The ResultsType contains the evaluation results for all OVAL Definitions on all systems under test.

Property	Type	Count	Description
results	SystemType	1..*	The evaluation results for all OVAL Definitions on each system under test.

Table 5: ResultsType Construct

7. SystemType

The SystemType provides the evaluation results for the OVAL Definitions and OVAL Tests as well the OVAL System Characteristics for an individual system.

Property	Type	Count	Description
definitions	DefinitionType	0..*	The evaluation results of the OVAL Definitions.
tests	TestType	0..*	The evaluation results of the OVAL Tests.
system_characteristics	oval-sc:oval_system_characteristics	1	A copy of the OVAL System Characteristics that were evaluated against the OVAL Definitions to produce the OVAL Results.

Table 6: SystemType Construct

8. DefinitionType

The DefinitionType contains the results of the evaluation of an OVAL Definition.

Property	Type	Count	Description
definition_id	oval:DefinitionIDPatterN	1	The unique identifier of

			an OVAL Definition that was used to generate the OVAL Results.
version	unsigned int	1	The version of the globally unique OVAL Definition.
variable_instance	unsigned int	0..1	The unique identifier that differentiates between each unique instance of an OVAL Definition. If an OVAL Definition utilizes an OVAL Variable, a unique instance of each OVAL Definition must be created for each collection of values assigned to the OVAL Variable. Default Value: '1'
class	oval:ClassEnumeration	0..1	The class of the OVAL Definition.
result	ResultEnumeration	1	The result of the evaluation of the OVAL Definition.

message	oval:MessageType	0..*	Any messages that are relayed from a tool at run-time during the evaluation of an OVAL Definition.
criteria	CriteriaType	0..1	Contains the individual results of the logical statements that form the OVAL Definition.

Table 7: DefinitionType Construct

9. CriteriaType

The CriteriaType combines the logical statements that form the OVAL Definition.

Property	Type	Count	Description
operator	oval:OperatorEnumeration	1	The logical operator that is used to combine the individual results of the logical statements defined by the child_criteria property.
negate	boolean	0..1	Specifies whether or not the evaluation result of the

				OVAL Definition, referenced by the definition_ref property, should be negated. Default Value: 'false'
result	ResultEnumeration	1		The evaluation result after the operator property and negate property have been applied.
criteria	CriteriaType	1..*		Logical statements that will be combined according to the operator property.
applicability_check	boolean	0..1		A boolean flag that when 'true' indicates that the criteria is being used to determine whether the OVAL Definition applies to a given system. No additional meaning is assumed when 'false'.

Table 8: CriteriaType Construct

10. CriterionType

The CriterionType is a logical statement that references an OVAL Test from an OVAL Definition.

Property	Type	Count	Description
test_ref	oval:TestIDPattern	1	The unique identifier of an OVAL Test contained in the OVAL Definitions used to generate the OVAL Results.
version	unsigned int	1	The version of the globally unique OVAL Test referenced by the test_ref property.
variable_instance	unsigned int	0..1	The unique identifier that differentiates between each unique instance of an OVAL Test. If an OVAL Test utilizes an OVAL Variable, a unique instance of each OVAL Test must be created for each collection of values assigned to

				the OVAL Variable. Default Value: '1'
negate	boolean	0..1		Specifies whether or not the evaluation result of the OVAL Test, referenced by the test_ref property, should be negated. Default Value: 'false'
result	ResultEnumeration	1		The evaluation result of the OVAL Test, referenced by the test_ref property, after the negate property has been applied.
applicability_check	boolean	0..1		A boolean flag that when true indicates that the criterion is being used to determine whether the OVAL Definition applies to a given system. No additional meaning is assumed when 'false'.

Table 9: CriterionType Construct

11. ExtendDefinitionType

The ExtendDefinitionType is a logical statement that references another OVAL Definition.

Property	Type	Count	Description
definition_ref	oval:DefinitionIDPattern	1	The unique identifier of an OVAL Definition used to generate the OVAL Results.
version	unsigned int	1	The version of the globally unique OVAL Definition referenced by the definition_ref property.
variable_instance	unsigned int	0..1	The unique identifier that differentiates between each unique instance of an OVAL Definition. If an OVAL Definition utilizes an OVAL Variable, a unique instance of each OVAL Definition must be created for each collection of values assigned to

				the OVAL Variable. Default Value: '1'
negate	boolean	0..1		Specifies whether or not the evaluation result of the OVAL Definition, referenced by the definition_ref property, should be negated. Default Value: 'false'
result	ResultEnumeration	1		The evaluation result of the OVAL Definition, referenced by the definition_ref property, after the negate property has been applied.
applicability_check	boolean	0..1		A boolean flag that when true indicates that the ExtendDefinition is being used to determine whether the OVAL Definition applies to a given system. No additional meaning is assumed when

				'false'.

Table 10: ExtendDefinitionType Construct

12. TestType

The TestType contains the result of an OVAL Test.

Property	Type	Count	Description
test_id	oval:TestIDPattern	1	The unique identifier of an OVAL Test contained in the OVAL Definitions used to generate the OVAL Results.
version	unsigned int	1	The version of the globally unique OVAL Test referenced by the test_id property.
variable_instance	unsigned int	0..1	The unique identifier that differentiates between each unique instance of an OVAL Test. If an OVAL Test utilizes an OVAL Variable, a unique instance of each OVAL Test must be created for each collection of values assigned to the OVAL Variable. Default Value: '1'
check_existence	oval:ExistenceEnumeration	0..1	Specifies how many OVAL Items must exist, on the system, in order for the OVAL Test

			to evaluate to true. Default Value: 'at_least_one_exists'
check	oval:CheckEnumeration	1	Specifies how many of the collected OVAL Items must satisfy the requirements specified by the OVAL State(s) in order for the OVAL Test to evaluate to true.
state_operator	oval:OperatorEnumeration	0..1	Specifies how to logically combine the OVAL States referenced in the OVAL Test. Default Value: 'AND'
result	ResultEnumeration	1	The evaluation result of the OVAL Test referenced by the test_id property.
message	oval:MessageType	0..*	Any messages that are relayed from a tool at run-time during the evaluation of an OVAL Test.
tested_item	TestedItemType	0..*	Specifies a reference to each OVAL Item used in the evaluation of an OVAL Test.
tested_variable	TestedVariableType	0..*	Specifies each OVAL Variable value used in the evaluation of an OVAL Test. This

				includes the OVAL Variable values used in both OVAL Objects and OVAL States.

Table 11: TestType Construct

13. TestedItemType

The TestedItemType contains the result of evaluating a collected OVAL Item against the OVAL State(s), if any, as specified by the corresponding OVAL Test.

Property	Type	Count	Description
item_id	oval:ItemIDPattern	1	The unique identifier of an OVAL Item collected during OVAL Item Collection.
result	ResultEnumeration	1	The evaluation result of the OVAL Item against the OVAL State(s), if any, as specified by the corresponding OVAL Test.
message	oval:MessageType	0..*	Any messages that are relayed from a tool at run-time during the evaluation of an OVAL Item against an OVAL State.

Table 12: TestedItemType Construct

14. TestedVariableType

The TestedVariableType specifies the value of an OVAL Variable used during the evaluation of an OVAL Test.

Property	Type	Count	Description
variable_id	oval:VariableIDPattern	1	The unique identifier of an OVAL Variable.
value	Any	1	A value of the OVAL Variable referenced by the variable_id property.

Table 13: TestedVariableType Construct

15. ContentEnumeration

The ContentEnumeration defines the acceptable levels of detail for the result information included in the OVAL Results.

Value	Description
thin	This value indicates that only the minimal amount of information is represented in the OVAL Results. The minimal set of information includes the following. The definition_id property of DefinitionType will be included. The result property of DefinitionType will be included. The criteria property of DefinitionType will not be included. The collected_objects and system_data properties, of the system_characteristics property in SystemType, will not be included.
full	This value indicates that a full detailed result of information is represented in the OVAL Results. The minimal set of information includes the following. The definition_id property of DefinitionType will be included. The result property of DefinitionType will be included. The criteria property of DefinitionType will be included. The collected_objects and system_data properties, of the system_characteristics property in SystemType, will be included. The value 'full' is equivalent to 'thin' with the collected_objects and system_data properties, of the system_characteristics property in SystemType, included.

Table 14: ContentEnumeration Construct

16. ResultEnumeration

The ResultEnumeration defines the acceptable evaluation result values in the OVAL Language.

Value	Description
true	This value indicates that the conditions of the evaluation were satisfied.
false	This value indicates that the conditions of the evaluation were not satisfied.
unknown	This value indicates that it could not be determined if the conditions of the evaluation were satisfied.
error	This value indicates that an error occurred during the evaluation.
not evaluated	This value indicates that a choice was made not to perform the evaluation.
not applicable	This value indicates that the evaluation being performed does not apply to the given platform.

Table 15: ResultEnumeration Construct

17. OVAL Results Model Schema

The XML Schema that implements this OVAL Results Model can be found below.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
  xmlns:oval-sc="http://oval.mitre.org/XMLSchema/
  oval-system-characteristics-5"
  xmlns:oval-def="http://oval.mitre.org/XMLSchema/
  oval-definitions-5"
  xmlns:oval-res="http://oval.mitre.org/XMLSchema/
  oval-results-5"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  targetNamespace="http://oval.mitre.org/XMLSchema/
  oval-results-5"
  elementFormDefault="qualified" version="5.11">
  <xsd:import
    namespace="http://oval.mitre.org/XMLSchema/
    oval-common-5"
```

```
      schemaLocation="oval-common-schema.xsd"/>
    <xsd:import
      namespace="http://oval.mitre.org/XMLSchema/
      oval-definitions-5"
      schemaLocation="oval-definitions-schema.xsd"/>
    <xsd:import
      namespace="http://oval.mitre.org/XMLSchema/
      oval-system-characteristics-5"
      schemaLocation="oval-system-characteristics-schema.xsd"/>
    <xsd:import
      namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="xmldsig-core-schema.xsd"/>
    <xsd:annotation>
      <xsd:documentation>The following is a
        description of the elements, types, and
        attributes that compose the core schema for
        encoding Open Vulnerability and Assessment
        Language (OVAL) Results. Each of the
        elements, types, and attributes that make up
        the Core Results Schema are described in
        detail and should provide the information
        necessary to understand what each object
        represents. This document is intended for
        developers and assumes some familiarity with
        XML. A high level description of the
        interaction between these objects is not
        outlined here.</xsd:documentation>
      <xsd:appinfo>
        <schema>Core Results</schema>
        <version>5.11.1</version>
        <date>4/22/2015 09:00:00 AM</date>
        <terms_of_use>Copyright (C) 2010 United States Government.
          All Rights Reserved.</terms_of_use>
        <sch:ns prefix="oval-res"
          uri="http://oval.mitre.org/XMLSchema/oval-results-5"
        />
      </xsd:appinfo>
    </xsd:annotation>
  <!-- ======================================================== -->
  <!-- ======================================================== -->
  <!-- ======================================================== -->
    <xsd:element name="oval_results">
      <xsd:annotation>
        <xsd:documentation>The oval_results element
          is the root of an OVAL Results Document.
          Its purpose is to bind together the four
          major sections of a results document -
          generator, directives, oval_definitions,
```

```
            and results - which are the children of
            the root element. It must contain exactly
            one generator section, one directives
            section, and one results
            section.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="generator"
              type="oval:GeneratorType">
              <xsd:annotation>
                <xsd:documentation>The required
                  generator section provides
                  information about when the results
                  document was compiled and under what
                  version.</xsd:documentation>
              </xsd:annotation>
            </xsd:element>
            <xsd:element name="directives"
              type="oval-res:DefaultDirectivesType">
              <xsd:annotation>
                <xsd:documentation>The required
                  directives section presents flags
                  describing what information has been
                  included in the results document.
                  This element represents the default
                  set of directives. These directives
                  apply to all classes of definitions
                  for which there is not a class
                  specific set of
                  directives.</xsd:documentation>
                <xsd:appinfo>
                  <sch:pattern
                    id="oval-res_directives_
                    include_oval_definitions">
                    <sch:rule
                      context="oval-res:oval_results/
                      oval-res:directives
                      [@include_source_definitions='true' or
                      @include_source_definitions='1' or
                      not(@include_source_definitions)]">
                      <sch:assert
                        test="ancestor::oval-res:oval_results
                        [oval-def:oval_definitions]"
                        > The source OVAL Definition
                        document must be included when
                        the directives
                        include_source_definitions
```

```
                    attribute is set to true.
                  </sch:assert>
                </sch:rule>
                <sch:rule
                  context="oval-res:oval_results/
                  oval-res:directives
                  [@include_source_definitions='false' or
                  @include_source_definitions='0']">
                  <sch:assert
                    test="ancestor::oval-res:oval_results
                    [not(oval-def:oval_definitions)]"
                    > The source OVAL Definition
                    document must not be included
                    when the directives
                    include_source_definitions
                    attribute is set to false.
                  </sch:assert>
                </sch:rule>
              </sch:pattern>
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="class_directives"
          type="oval-res:ClassDirectivesType"
          minOccurs="0" maxOccurs="5">
          <xsd:annotation>
            <xsd:documentation>The optional
              class_directives section presents
              flags describing what information
              has been included in the results
              document for a specific OVAL
              Definition class. The directives for
              a particlar class override the
              default directives. Using OVAL
              Results class_directives, an OVAL
              Results document dealing with
              vulnerabilities might by default
              include only minimal information and
              then include full details for all
              vulnerability definitions that
              evaluated to
              true.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element
          ref="oval-def:oval_definitions"
          minOccurs="0" maxOccurs="1">
          <xsd:annotation>
```

```
            <xsd:documentation>The
              oval_definitions section is optional
              and dependent on the
              include_source_definitions attribute
              of the directives element. Its
              purpose is to provide an exact copy
              of the definitions evaluated for the
              results
              document.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="results"
          type="oval-res:ResultsType">
          <xsd:annotation>
            <xsd:documentation>The required
              results section holds all the
              results of the evaluated
              definitions.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element ref="ds:Signature"
          minOccurs="0" maxOccurs="1">
          <xsd:annotation>
            <xsd:documentation>The optional
              Signature element allows an XML
              Signature as defined by the W3C to
              be attached to the document. This
              allows authentication and data
              integrity to be provided to the
              user. Enveloped signatures are
              supported. More information about
              the official W3C Recommendation
              regarding XML digital signatures can
              be found at
              http://www.w3.org/TR/xmldsig-core/.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="UniqueDirectiveClass">
      <xsd:annotation>
        <xsd:documentation>The class attribute on
          class_directives must be
          unique.</xsd:documentation>
      </xsd:annotation>
      <xsd:selector
        xpath="oval-res:class_directives"/>
```

```
        <xsd:field xpath="@class"/>
      </xsd:unique>
    </xsd:element>

  <!-- ======================================================== -->
  <!-- ==================  GENERATOR  ======================== -->
  <!-- ======================================================== -->
    <!--
                The GeneratorType is defined by the
                oval-common-schema.  Please refer to
                that documentation for a description
                of the complex type.
          -->
  <!-- ======================================================== -->
  <!-- ==================  DIRECTIVES  ======================== -->
  <!-- ======================================================== -->
    <xsd:complexType name="DirectivesType">
      <xsd:annotation>
        <xsd:documentation>The DirectivesType
          complex type presents a set of flags that
          describe what information has been
          included in the results document. There
          are six possible results (true, false,
          unknown, error, not evaluated, and not
          applicable) for the evaluation of an OVAL
          Definition. The directives state which of
          these results are being reported in the
          results document.</xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="definition_true"
          type="oval-res:DirectiveType"/>
        <xsd:element name="definition_false"
          type="oval-res:DirectiveType"/>
        <xsd:element name="definition_unknown"
          type="oval-res:DirectiveType"/>
        <xsd:element name="definition_error"
          type="oval-res:DirectiveType"/>
        <xsd:element name="definition_not_evaluated"
          type="oval-res:DirectiveType"/>
        <xsd:element
          name="definition_not_applicable"
          type="oval-res:DirectiveType"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DefaultDirectivesType">
      <xsd:annotation>
        <xsd:documentation>The DefaultDirectivesType
```

```
          complex type presents the default set of
          flags that describe what information has
          been included in the results document. See
          the definition of the
          oval-res:DirectivesType for more
          information.</xsd:documentation>
        <xsd:documentation>The optional
          include_source_definitions attribute
          indicates whether or not the source OVAL
          Definitions document has been included in
          the results document. A value of false
          indicates that the source OVAL Definitions
          has not been included. By default the
          source document is
          included.</xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
        <xsd:extension
          base="oval-res:DirectivesType">
          <xsd:attribute
            name="include_source_definitions"
            type="xsd:boolean" default="true"
            use="optional"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ClassDirectivesType">
      <xsd:annotation>
        <xsd:documentation>The ClassDirectivesType
          complex type presents a set of flags that
          describe what information has been
          included in the results document for a
          specific OVAL Definition class. See the
          definition of the oval-res:DirectivesType
          for more information.</xsd:documentation>
        <xsd:documentation>The required class
          attribute allows a set of directives to be
          specified for each supported OVAL
          Definition class (See the definition of
          the oval:ClassEnumeration for more
          information about the supported classes).
          A set of class specific directives
          overrides the default directives for the
          specified definition class. A given class
          may be specified once.</xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
        <xsd:extension
```

```
          base="oval-res:DirectivesType">
          <xsd:attribute name="class"
            type="oval:ClassEnumeration"
            use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="DirectiveType">
      <xsd:annotation>
        <xsd:documentation>An individual directive
          element determines whether or not a
          specific type of result is included in the
          results document. The required reported
          attribute controls this by providing a
          true or false for the specific directive.
          The optional content attribute controls
          how much information about the specific
          result is provided. For example, thin
          content would only be the id of the
          definition and the result, while a full
          content set would be the definition id
          with the result along with results for all
          the individual tests and extended
          definitions. Please refer to the
          oval-res:ContentEnumeration for details
          about the different content
          options.</xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="reported"
        type="xsd:boolean" use="required"/>
      <xsd:attribute name="content"
        type="oval-res:ContentEnumeration"
        use="optional" default="full"/>
    </xsd:complexType>
  <!-- ======================================================= -->
  <!-- ===============  OVAL DEFINITIONS  ==================== -->
  <!-- ======================================================= -->
  <!--
          The oval_definitions element is defined by the oval
          definitions schema.  Please refer to that
          documentation for a description of the valid elements
          and types.
   -->
  <!-- ======================================================= -->
  <!-- ====================  RESULTS  ======================== -->
  <!-- ======================================================= -->
    <xsd:complexType name="ResultsType">
      <xsd:annotation>
```

```
          <xsd:documentation>The ResultsType complex
            type is a container for one or more system
            elements. Each system element defines the
            results associated with an individual
            system. Please refer to the description of
            SystemType for more information about an
            individual system
            element.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
          <xsd:element name="system"
            type="oval-res:SystemType" minOccurs="1"
            maxOccurs="unbounded">
            <xsd:key name="definitionInstanceKey">
              <xsd:annotation>
                <xsd:documentation>Enforce uniqueness
                  in the combination of OVAL id,
                  version, and variable_instance in
                  order to differentiate the
                  individual definition
                  elements.</xsd:documentation>
              </xsd:annotation>
              <xsd:selector
                xpath="oval-res:definitions/oval-res:definition"/>
              <xsd:field xpath="@definition_id"/>
              <xsd:field xpath="@version"/>
              <xsd:field xpath="@variable_instance"/>
            </xsd:key>
            <xsd:key name="testVersionKey">
              <xsd:annotation>
                <xsd:documentation>Enforce uniqueness
                  in the combination of the individual
                  test ids, version, and the
                  variable_instance of the
                  test.</xsd:documentation>
              </xsd:annotation>
              <xsd:selector
                xpath="oval-res:tests/oval-res:test"/>
              <xsd:field xpath="@test_id"/>
              <xsd:field xpath="@version"/>
              <xsd:field xpath="@variable_instance"/>
            </xsd:key>
            <xsd:keyref
              name="definitionInstanceKeyRef"
              refer="oval-res:definitionInstanceKey">
              <xsd:annotation>
                <xsd:documentation>Requires each
                  definition reference (used by
```

```
                extend_definitions) to refer to a
                valid definition
                id.</xsd:documentation>
            </xsd:annotation>
            <xsd:selector xpath=".//*"/>
            <xsd:field xpath="@definition_ref"/>
            <xsd:field xpath="@version"/>
            <xsd:field xpath="@variable_instance"/>
          </xsd:keyref>
          <xsd:keyref name="testVersionKeyRef"
            refer="oval-res:testVersionKey">
            <xsd:annotation>
              <xsd:documentation>Requires each test
                reference to refer to a valid test
                id.</xsd:documentation>
            </xsd:annotation>
            <xsd:selector xpath=".//*"/>
            <xsd:field xpath="@test_ref"/>
            <xsd:field xpath="@version"/>
            <xsd:field xpath="@variable_instance"/>
          </xsd:keyref>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="SystemType">
      <xsd:annotation>
        <xsd:documentation>The SystemType complex
          type holds the evaluation results of the
          definitions and tests, as well as a copy
          of the OVAL System Characteristics used to
          perform the evaluation. The definitions
          section holds the results of the
          definitions and the tests section holds
          the results of the tests. The
          oval_system_characteristics section is a
          copy of the System Characteristics
          document used to perform the evaluation of
          the OVAL Definitions.</xsd:documentation>
        <xsd:appinfo>
          <sch:pattern id="oval-res_system">
            <sch:rule
              context="oval-res:system[oval-res:tests]">
              <!-- Confirm that something somewhere
                expects full results -->
              <sch:assert
                test="/oval-res:oval_results/
                oval-res:directives/*[@reported='true' or
                @reported='1']/@content='full'
```

```
                or /oval-res:oval_results/
                oval-res:directives/*[(@reported='true' or
                @reported='1') and not(@content)] or
                /oval-res:oval_results/
                oval-res:class_directives/*
                [@reported='true' or @reported='1']/
                @content='full' or /oval-res:oval_results/
                oval-res:class_directives/*
                [(@reported='true' or @reported='1') and
                not(@content)]"
                > The tests element should not be
                included unless full results are to
                be provided (see directives)
              </sch:assert>
            </sch:rule>
            <sch:rule
              context="oval-res:system
              [not(oval-res:tests)]">
              <!-- Confirm that nothing anywhere
                expects full results -->
              <sch:assert
                test="not(oval-res:oval_results/
                oval-res:directives/*
                [@reported='true' or @reported='1']/
                @content='full') and
                not(/oval-res:oval_results/
                oval-res:directives/*
                [(@reported='true' or @reported='1') and
                not(@content)]) and
                not(/oval-res:oval_results/
                oval-res:class_directives/*
                [@reported='true' or
                @reported='1']/@content='full') and
                not(/oval-res:oval_results/
                oval-res:class_directives/*
                [(@reported='true' or
                @reported='1') and
                not(@content)])"
                > The tests element should be
                included when full results are
                specified (see directives)
              </sch:assert>
            </sch:rule>
          </sch:pattern>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="definitions"
```

```
          type="oval-res:DefinitionsType"
          minOccurs="0" maxOccurs="1"/>
        <xsd:element name="tests"
          type="oval-res:TestsType" minOccurs="0"
          maxOccurs="1"/>
        <xsd:element
          ref="oval-sc:oval_system_characteristics">
          <xsd:annotation>
            <xsd:appinfo>
              <sch:pattern id="oval-res_mask_rule">
                <sch:rule
                  context="/oval-res:oval_results/
                  oval-res:results/oval-res:system/
                  oval-sc:oval_system_characteristics/
                  oval-sc:system_data/*/*|
                  /oval-res:oval_results/oval-res:results/
                  oval-res:system/
                  oval-sc:oval_system_characteristics/
                  oval-sc:system_data/*/*/*">
                  <sch:assert
                    test="not(@mask) or
                    @mask='false' or
                    @mask='0' or .=''"
                    >item <sch:value-of
                      select="../@id"/> - a value
                    for the <sch:value-of
                      select="name()"/> entity
                    should only be supplied if the
                    mask attribute is
                    'false'.</sch:assert>
                </sch:rule>
              </sch:pattern>
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DefinitionsType">
      <xsd:annotation>
        <xsd:documentation>The DefinitionsType
          complex type is a container for one or
          more definition elements. Each definition
          element holds the result of the evaluation
          of an OVAL Definition. Please refer to the
          description of DefinitionType for more
          information about an individual definition
          element.</xsd:documentation>
      </xsd:annotation>
```

```
      <xsd:sequence>
        <xsd:element name="definition"
          type="oval-res:DefinitionType"
          minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="DefinitionType">
      <xsd:annotation>
        <xsd:documentation>The DefinitionType
          complex type holds the result of the
          evaluation of an OVAL Definition. The
          message element holds an error message or
          some other string that the analysis engine
          wishes to pass along. In addition, the
          optional criteria element provides the
          results of the individual pieces of the
          criteria. Please refer to the description
          of the CriteriaType for more
          information.</xsd:documentation>
        <xsd:documentation>The required
          definition_id attribute is the OVAL id of
          the definition.</xsd:documentation>
        <xsd:documentation>The required version
          attribute is the specific version of the
          OVAL Definition used during
          analysis.</xsd:documentation>
        <xsd:documentation>The optional
          variable_instance attribute is a unique id
          that differentiates each unique instance
          of a definition. Capabilities that use
          OVAL may reference the same definition
          multiple times and provide different
          variable values each time the definition
          is referenced. This will result in
          multiple instances of a definition being
          included in the OVAL Results document
          (definitions that do not use variables can
          only have one unique instance). The
          inclusion of this unique instance
          identifier allows the OVAL Results
          document to associate the correct objects
          and items for each combination of supplied
          values.</xsd:documentation>
        <xsd:documentation>The optional class
          attribute ...</xsd:documentation>
        <xsd:documentation>The required result
          attribute holds the result of the
          evaluation. Please refer to the
```

```
            description of the ResultEnumeration for
            details about the different result
            values.</xsd:documentation>
          <xsd:appinfo>
            <sch:pattern id="oval-res_directives">
              <!-- Check definition_true
                reported='true' and content='full' -->
              <sch:rule
                context="oval-res:definition
                [@result='true' and oval-res:criteria]">
                <!-- Check that the global directives say to
                  report this and that there are no class
                  directives for this class (to override
                  the global directive), or that the class
                  directive for this class says to report
                  this. -->
                <sch:assert
                  test="((/oval-res:oval_results/
                  oval-res:directives/
                  oval-res:definition_true/
                  @reported='true' or
                  /oval-res:oval_results/
                  oval-res:directives/
                  oval-res:definition_true/
                  @reported='1') and
                  not(oval-res:oval_results/
                  oval-res:class_directives
                  [@class = ./@class])) or
                  (/oval-res:oval_results/
                  oval-res:class_directives
                  [@class = ./@class]/
                  oval-res:definition_true/
                  @reported='true' or
                  /oval-res:oval_results/
                  oval-res:class_directives
                  [@class = ./@class]/
                  oval-res:definition_true/
                  @reported='1')">
                  <sch:value-of
                    select="@definition_id"/> -
                  definitions with a result of TRUE
                  should not be included (see
                  directives) </sch:assert>
                <sch:assert
                  test="((/oval-res:oval_results/
                  oval-res:directives/
                  oval-res:definition_true/
                  @content='full') and
```

```
      not(/oval-res:oval_results/
      oval-res:class_directives
      [@class = ./@class])) or
      (/oval-res:oval_results/
      oval-res:class_directives
      [@class = ./@class]/
      oval-res:definition_true/
      @content='full')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of TRUE
      should contain THIN content (see
      directives) </sch:assert>
    </sch:rule>

    <!-- Check definition_true
      reported='true' and content='thin' -->
    <sch:rule
      context="oval-res:definition[@result='true'
      and not(oval-res:criteria)]">
      <sch:assert
        test="((/oval-res:oval_results/
        oval-res:directives/
        oval-res:definition_true/
        @reported='true' or
        /oval-res:oval_results/
        oval-res:directives/
        oval-res:definition_true/
        @reported='1') and not
        (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class])) or
        (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class]/
        oval-res:definition_true/
        @reported='true' or
        /oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class]/
        oval-res:definition_true/@reported='1')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of TRUE
        should not be included (see
        directives) </sch:assert>
      <sch:assert
        test="((/oval-res:oval_results/
```

```
        oval-res:directives/
        oval-res:definition_true/@content='thin')
        and not(/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class])) or
        (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class]/
        oval-res:definition_true/
        @content='thin')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of TRUE
        should contain FULL content (see
        directives) </sch:assert>
      </sch:rule>

      <!-- Check definition_false reported='true'
        and content='full' -->
      <sch:rule
        context="oval-res:definition[@result='false'
        and oval-res:criteria]">
        <sch:assert
          test="((/oval-res:oval_results/
          oval-res:directives/
          oval-res:definition_false/
          @reported='true' or
          /oval-res:oval_results/
          oval-res:directives/
          oval-res:definition_false/
          @reported='1') and
          not(/oval-res:oval_results/
          oval-res:class_directives[@class = ./@class]))
          or (/oval-res:oval_results/
          oval-res:class_directives
          [@class = ./@class]/
          oval-res:definition_false/
          @reported='true' or
          /oval-res:oval_results/
          oval-res:class_directives
          [@class = ./@class]/
          oval-res:definition_false/@reported='1')">
          <sch:value-of
            select="@definition_id"/> -
          definitions with a result of FALSE
          should not be included (see
          directives) </sch:assert>
        <sch:assert
```

```
      test="((/oval-res:oval_results/
      oval-res:directives/
      oval-res:definition_false/@content='full')
      and not(/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_false/@content='full')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of FALSE
      should contain THIN content (see
      directives) </sch:assert>
    </sch:rule>

    <!-- Check definition_false reported='true'
      and content='thin' -->
    <sch:rule
      context="oval-res:definition[@result='false' and
      not(oval-res:criteria)]">
      <sch:assert
        test="((/oval-res:oval_results/
        oval-res:directives/oval-res:definition_false/
        @reported='true' or /oval-res:oval_results/
        oval-res:directives/oval-res:definition_false/
        @reported='1') and not(/oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]))
        or (/oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]/
        oval-res:definition_false/@reported='true' or
        /oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]/
        oval-res:definition_false/@reported='1')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of FALSE
        should not be included (see
        directives) </sch:assert>
      <sch:assert
        test="((/oval-res:oval_results/
        oval-res:directives/oval-res:definition_false/
        @content='thin') and not(/oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]))
        or (/oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]/
        oval-res:definition_false/@content='thin')">
        <sch:value-of
          select="@definition_id"/> -
```

```
        definitions with a result of FALSE
        should contain FULL content (see
        directives) </sch:assert>
    </sch:rule>

    <!-- Check definition_unknown reported='true'
      and content='full' -->
    <sch:rule
      context="oval-res:definition[@result='unknown'
      and oval-res:criteria]">
      <sch:assert
        test="((/oval-res:oval_results/
        oval-res:directives/
        oval-res:definition_unknown/
        @reported='true' or /oval-res:oval_results/
        oval-res:directives/
        oval-res:definition_unknown/@reported='1')
        and not(/oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]))
        or (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class]/
        oval-res:definition_unknown/@reported='true'
        or /oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]/
        oval-res:definition_unknown/@reported='1')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of UNKNOWN
        should not be included (see
        directives) </sch:assert>
      <sch:assert
        test="((/oval-res:oval_results/
        oval-res:directives/oval-res:definition_unknown/
        @content='full') and not(oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]))
        or (/oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]/
        oval-res:definition_unknown/@content='full')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of UNKNOWN
        should contain THIN content (see
        directives) </sch:assert>
    </sch:rule>

    <!-- Check definition_unknown reported='true'
      and content='thin' -->
```

```
            <sch:rule
              context="oval-res:definition[@result='unknown'
              and not(oval-res:criteria)]">
              <sch:assert
                test="((/oval-res:oval_results/
                oval-res:directives/oval-res:definition_unknown/
                @reported='true' or /oval-res:oval_results/
                oval-res:directives/
                oval-res:definition_unknown/@reported='1')
                and not(/oval-res:oval_results/
                oval-res:class_directives[@class = ./@class]))
                or (/oval-res:oval_results/
                oval-res:class_directives[@class = ./@class]/
                oval-res:definition_unknown/@reported='true' or
                /oval-res:oval_results/
                oval-res:class_directives[@class = ./@class]/
                oval-res:definition_unknown/@reported='1')">
                <sch:value-of
                  select="@definition_id"/> -
                definitions with a result of UNKNOWN
                should not be included (see
                directives) </sch:assert>
              <sch:assert
                test="((/oval-res:oval_results/
                oval-res:directives/
                oval-res:definition_unknown/@content='thin')
                and not(/oval-res:oval_results/
                oval-res:class_directives[@class = ./@class]))
                or (/oval-res:oval_results/
                oval-res:class_directives
                [@class = ./@class]/oval-res:definition_unknown/
                @content='thin')">
                <sch:value-of
                  select="@definition_id"/> -
                definitions with a result of UNKNOWN
                should contain FULL content (see
                directives) </sch:assert>
            </sch:rule>

            <!-- Check definition_error reported='true'
              and content='full' -->
            <sch:rule
              context="oval-res:definition[@result='error'
              and oval-res:criteria]">
              <sch:assert
                test="((/oval-res:oval_results/
                oval-res:directives/oval-res:definition_error/
                @reported='true' or /oval-res:oval_results/
```

```
      oval-res:directives/oval-res:definition_error/
      @reported='1') and not(/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_error/@reported='true' or
      /oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_error/@reported='1')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of ERROR
      should not be included (see
      directives) </sch:assert>
    <sch:assert
      test="((/oval-res:oval_results/
      oval-res:directives/oval-res:definition_error/
      @content='full') and not(oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_error/@content='full')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of ERROR
      should contain THIN content (see
      directives) </sch:assert>
  </sch:rule>

  <!-- Check definition_error reported='true' and
    content='thin' -->
  <sch:rule
    context="oval-res:definition[@result='error'
    and not(oval-res:criteria)]">
    <sch:assert
      test="((/oval-res:oval_results/
      oval-res:directives/oval-res:definition_error/
      @reported='true' or /oval-res:oval_results/
      oval-res:directives/oval-res:definition_error/
      @reported='1') and not(/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_unknown/
      @reported='true' or /oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_unknown/@reported='1')">
      <sch:value-of
```

```
                select="@definition_id"/> -
              definitions with a result of ERROR
              should not be included (see
              directives) </sch:assert>
            <sch:assert
              test="((/oval-res:oval_results/
              oval-res:directives/oval-res:definition_error/
              @content='thin') and not(/oval-res:oval_results/
              oval-res:class_directives[@class = ./@class]))
              or (/oval-res:oval_results/
              oval-res:class_directives[@class = ./@class]/
              oval-res:definition_error/@content='thin')">
              <sch:value-of
                select="@definition_id"/> -
              definitions with a result of ERROR
              should contain FULL content (see
              directives) </sch:assert>
          </sch:rule>

          <!-- Check definition_not_evaluated
            reported='true' and content='full' -->
          <sch:rule
            context="oval-res:definition[@result='not evaluated'
            and oval-res:criteria]">
            <sch:assert
              test="((/oval-res:oval_results/
              oval-res:directives/
              oval-res:definition_not_evaluated/
              @reported='true' or /oval-res:oval_results/
              oval-res:directives/
              oval-res:definition_not_evaluated/@reported='1')
              and not(/oval-res:oval_results/
              oval-res:class_directives[@class = ./@class]))
              or (/oval-res:oval_results/
              oval-res:class_directives[@class = ./@class]/
              oval-res:definition_not_evaluated/
              @reported='true' or /oval-res:oval_results/
              oval-res:class_directives[@class = ./@class]/
              oval-res:definition_not_evaluated/
              @reported='1')">
              <sch:value-of
                select="@definition_id"/> -
              definitions with a result of NOT
              EVALUATED should not be included
              (see directives) </sch:assert>
            <sch:assert
              test="((/oval-res:oval_results/
              oval-res:directives/
```

```
      oval-res:definition_not_evaluated/
      @content='full') and not
      (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class =
      ./@class]/oval-res:definition_not_evaluated/
      @content='full')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of NOT
      EVALUATED should contain THIN
      content (see directives)
    </sch:assert>
  </sch:rule>

  <!-- Check definition_not_evaluated
    reported='true' and content='thin' -->
  <sch:rule
    context="oval-res:definition
    [@result='not evaluated' and
    not(oval-res:criteria)]">
    <sch:assert
      test="((/oval-res:oval_results/
      oval-res:directives/
      oval-res:definition_not_evaluated/
      @reported='true' or /oval-res:oval_results/
      oval-res:directives/
      oval-res:definition_not_evaluated/
      @reported='1') and not(/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class =
      ./@class]/oval-res:definition_not_evaluated/
      @reported='true' or /oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_not_evaluated/
      @reported='1')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of NOT
      EVALUATED should not be included
      (see directives) </sch:assert>
    <sch:assert
      test="((/oval-res:oval_results/
      oval-res:directives/
      oval-res:definition_not_evaluated/
      @content='thin') and not
```

```
      (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]))
      or (/oval-res:oval_results/
      oval-res:class_directives[@class = ./@class]/
      oval-res:definition_not_evaluated/
      @content='thin')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of NOT
      EVALUATED should contain FULL
      content (see directives)
    </sch:assert>
  </sch:rule>

  <!-- Check definition_not_applicable
    reported='true' and content='full' -->
  <sch:rule
    context="oval-res:definition
    [@result='not applicable' and
    oval-res:criteria]">
    <sch:assert
      test="((/oval-res:oval_results/
      oval-res:directives/
      oval-res:definition_not_applicable/
      @reported='true' or /oval-res:oval_results/
      oval-res:directives/
      oval-res:definition_not_applicable/
      @reported='1') and not
      (/oval-res:oval_results/
      oval-res:class_directives
      [@class = ./@class])) or
      (/oval-res:oval_results/
      oval-res:class_directives
      [@class = ./@class]/
      oval-res:definition_not_applicable/
      @reported='true' or
      /oval-res:oval_results/
      oval-res:class_directives
      [@class = ./@class]/
      oval-res:definition_not_applicable/
      @reported='1')">
      <sch:value-of
        select="@definition_id"/> -
      definitions with a result of NOT
      APPLICABLE should not be included
      (see directives) </sch:assert>
    <sch:assert
      test="((/oval-res:oval_results/
```

```
        oval-res:directives/
        oval-res:definition_not_applicable/
        @content='full') and not
        (oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]))
        or (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class]/
        oval-res:definition_not_applicable/
        @content='full')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of NOT
        APPLICABLE should contain THIN
        content (see directives)
      </sch:assert>
    </sch:rule>

    <!-- Check definition_not_applicable
      reported='true' and content='thin' -->
    <sch:rule
      context="oval-res:definition
      [@result='not applicable' and
      not(oval-res:criteria)]">
      <sch:assert
        test="((/oval-res:oval_results/
        oval-res:directives/
        oval-res:definition_not_applicable/
        @reported='true' or /oval-res:oval_results/
        oval-res:directives/
        oval-res:definition_not_applicable/
        @reported='1') and not
        (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class])) or
        (/oval-res:oval_results/
        oval-res:class_directives
        [@class = ./@class]/
        oval-res:definition_not_applicable/
        @reported='true' or /oval-res:oval_results/
        oval-res:class_directives[@class = ./@class]
        /oval-res:definition_not_applicable/
        @reported='1')">
        <sch:value-of
          select="@definition_id"/> -
        definitions with a result of NOT
        APPLICABLE should not be included
        (see directives) </sch:assert>
```

```
              <sch:assert
                test="((/oval-res:oval_results/
                oval-res:directives/
                oval-res:definition_not_applicable/
                @content='thin') and not
                (oval-res:oval_results/
                oval-res:class_directives
                [@class = ./@class])) or
                (/oval-res:oval_results/
                oval-res:class_directives
                [@class = ./@class]/
                oval-res:definition_not_applicable/
                @content='thin')">
                <sch:value-of
                  select="@definition_id"/> -
                definitions with a result of NOT
                APPLICABLE should contain FULL
                content (see directives)
              </sch:assert>
            </sch:rule>
          </sch:pattern>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="message"
          type="oval:MessageType" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="criteria"
          type="oval-res:CriteriaType" minOccurs="0"
          maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="definition_id"
        type="oval:DefinitionIDPattern"
        use="required"/>
      <xsd:attribute name="version"
        type="xsd:nonNegativeInteger" use="required"/>
      <xsd:attribute name="variable_instance"
        type="xsd:nonNegativeInteger" use="optional"
        default="1"/>
      <xsd:attribute name="class"
        type="oval:ClassEnumeration" use="optional"/>
      <xsd:attribute name="result"
        type="oval-res:ResultEnumeration"
        use="required"/>
    </xsd:complexType>
    <xsd:complexType name="CriteriaType">
      <xsd:annotation>
        <xsd:documentation>The CriteriaType complex
```

```
        type describes the high level container
        for all the tests and represents the meat
        of the definition. Each criteria can
        contain other criteria elements in a
        recursive structure allowing complex
        logical trees to be constructed. Each
        referenced test is represented by a
        criterion element. Please refer to the
        description of the CriterionType for more
        information about and individual criterion
        element. The optional extend_definition
        element allows existing definitions to be
        included in the criteria. Refer to the
        description of the ExtendDefinitionType
        for more information.</xsd:documentation>
      <xsd:documentation>The required operator
        attribute provides the logical operator
        that binds the different statements inside
        a criteria together. The optional negate
        attribute signifies that the result of an
        extended definition should be negated
        during analysis. For example, consider a
        definition that evaluates TRUE if a
        certain software is installed. By negating
        the definition, it now evaluates to TRUE
        if the software is NOT installed. The
        required result attribute holds the result
        of the evaluation of the criteria. Note
        that this would be after any negation
        operation has been applied. Please refer
        to the description of the
        ResultEnumeration for details about the
        different result
        values.</xsd:documentation>
      <xsd:documentation>The optional
        applicability_check attribute provides a
        Boolean flag that when true indicates that
        the criteria is being used to determine
        whether the OVAL Definition applies to a
        given system.</xsd:documentation>
    </xsd:annotation>
    <xsd:choice minOccurs="1"
      maxOccurs="unbounded">
      <xsd:element name="criteria"
        type="oval-res:CriteriaType"/>
      <xsd:element name="criterion"
        type="oval-res:CriterionType"/>
      <xsd:element name="extend_definition"
```

```
          type="oval-res:ExtendDefinitionType"/>
      </xsd:choice>
      <xsd:attribute name="applicability_check"
        type="xsd:boolean" use="optional"/>
      <xsd:attribute name="operator"
        type="oval:OperatorEnumeration"
        use="required"/>
      <xsd:attribute name="negate"
        type="xsd:boolean" use="optional"
        default="false"/>
      <xsd:attribute name="result"
        type="oval-res:ResultEnumeration"
        use="required"/>
    </xsd:complexType>
    <xsd:complexType name="CriterionType">
      <xsd:annotation>
        <xsd:documentation>The CriterionType complex
          type identifies a specific test that is
          included in the definition's
          criteria.</xsd:documentation>
        <xsd:documentation>The optional
          applicability_check attribute provides a
          Boolean flag that when true indicates that
          the criterion is being used to determine
          whether the OVAL Definition applies to a
          given system.</xsd:documentation>
        <xsd:documentation>The required test_ref
          attribute is the actual id of the included
          test.</xsd:documentation>
        <xsd:documentation>The required version
          attribute is the specific version of the
          OVAL Test used during
          analysis.</xsd:documentation>
        <xsd:documentation>The optional
          variable_instance attribute differentiates
          between unique instances of a test. This
          can happen when a test includes a variable
          reference and different variable values
          are used by different
          definitions.</xsd:documentation>
        <xsd:documentation>The optional negate
          attribute signifies that the result of an
          individual test should be negated during
          analysis. For example, consider a test
          that evaluates to TRUE if a specific patch
          is installed. By negating this test, it
          now evaluates to TRUE if the patch is NOT
          installed.</xsd:documentation>
```

```
        <xsd:documentation>The required result
          attribute holds the result of the
          evaluation. Please refer to the
          description of the ResultEnumeration for
          details about the different result
          values.</xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="applicability_check"
        type="xsd:boolean" use="optional"/>
      <xsd:attribute name="test_ref"
        type="oval:TestIDPattern" use="required"/>
      <xsd:attribute name="version"
        type="xsd:nonNegativeInteger" use="required"/>
      <xsd:attribute name="variable_instance"
        type="xsd:nonNegativeInteger" use="optional"
        default="1"/>
      <xsd:attribute name="negate"
        type="xsd:boolean" use="optional"
        default="false"/>
      <xsd:attribute name="result"
        type="oval-res:ResultEnumeration"
        use="required"/>
    </xsd:complexType>
    <xsd:complexType name="ExtendDefinitionType">
      <xsd:annotation>
        <xsd:documentation>The ExtendDefinitionType
          complex type identifies a specific
          definition that has been extended by the
          criteria.</xsd:documentation>
        <xsd:documentation>The optional
          applicability_check attribute provides a
          Boolean flag that when true indicates that
          the extend_definition is being used to
          determine whether the OVAL Definition
          applies to a given
          system.</xsd:documentation>
        <xsd:documentation>The required
          definition_ref attribute is the actual id
          of the extended
          definition.</xsd:documentation>
        <xsd:documentation>The required version
          attribute is the specific version of the
          OVAL Definition used during
          analysis.</xsd:documentation>
        <xsd:documentation>The optional
          variable_instance attribute is a unique id
          that differentiates each unique instance
          of a definition. Capabilities that use
```

```
          OVAL may reference the same definition
          multiple times and provide different
          variable values each time the definition
          is referenced. This will result in
          multiple instances of a definition being
          included in the OVAL Results document
          (definitions that do not use variables can
          only have one unique instance). The
          inclusion of this unique instance
          identifier allows the OVAL Results
          document to associate the correct objects
          and items for each combination of supplied
          values.</xsd:documentation>
        <xsd:documentation>The optional negate
          attribute signifies that the result of an
          extended definition should be negated
          during analysis. For example, consider a
          definition that evaluates TRUE if certain
          software is installed. By negating the
          definition, it now evaluates to TRUE if
          the software is NOT
          installed.</xsd:documentation>
        <xsd:documentation>The required result
          attribute holds the result of the
          evaluation. Please refer to the
          description of the ResultEnumeration for
          details about the different result
          values.</xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="applicability_check"
        type="xsd:boolean" use="optional"/>
      <xsd:attribute name="definition_ref"
        type="oval:DefinitionIDPattern"
        use="required"/>
      <xsd:attribute name="version"
        type="xsd:nonNegativeInteger" use="required"/>
      <xsd:attribute name="variable_instance"
        type="xsd:nonNegativeInteger" use="optional"
        default="1"/>
      <xsd:attribute name="negate"
        type="xsd:boolean" use="optional"
        default="false"/>
      <xsd:attribute name="result"
        type="oval-res:ResultEnumeration"
        use="required"/>
    </xsd:complexType>
    <xsd:complexType name="TestsType">
      <xsd:annotation>
```

```
        <xsd:documentation>The TestsType complex
          type is a container for one or more test
          elements. Each test element holds the
          result of the evaluation of an OVAL Test.
          Please refer to the description of
          TestType for more information about an
          individual test
          element.</xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="test"
          type="oval-res:TestType" minOccurs="1"
          maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TestType">
      <xsd:annotation>
        <xsd:documentation>The TestType complex type
          provides a reference to every item that
          matched the object section of the original
          test as well as providing an overall test
          result based on those items. The optional
          message element holds an error message or
          some other string that the analysis engine
          wishes to pass along. The optional
          tested_variable elements hold the value of
          each variable used by the test during
          evaluation. This includes the values used
          in both OVAL Objects and OVAL States. If a
          variable represents a collection of
          values, then multiple tested_variable
          elements would exist with the same
          variable_id attribute. Please refer to the
          description of oval-res:TestedVariableType
          for more information.</xsd:documentation>
        <xsd:documentation>The required test_id
          attribute identifies the test and must
          conform to the format specified by the
          oval:TestIDPattern simple
          type.</xsd:documentation>
        <xsd:documentation>The required version
          attribute is the specific version of the
          OVAL Test used during
          analysis.</xsd:documentation>
        <xsd:documentation>The optional
          variable_instance attribute differentiates
          between unique instances of a test. This
          can happen when a test includes a variable
```

```
        reference and different values for that
        variable are used by different
        definitions.</xsd:documentation>
      <xsd:documentation>The check_existence,
        check, and state_operator attributes
        reflect the values that were specified on
        the test as it was evaluated. These
        evaluation control attributes are copied
        into the OVAL Results file to enable post
        processing of results documents. More
        information on each of these attributes is
        provided with the definition of the
        oval-def:TestType.</xsd:documentation>
      <xsd:documentation>The required result
        attribute holds the result of the
        evaluation after all referenced items have
        been examined and the evaluation control
        attributes have been applied. Please refer
        to the description of the
        oval-res:ResultEnumeration for details
        about the different result values. In
        general, the overall result of an OVAL
        Test is determined by combining the
        results of each matching item based first
        on the check_existence attribute, then the
        check attribute, and finally the
        state_operator
        attribute.</xsd:documentation>
      <xsd:documentation>The following section
        provides a more detailed description of
        how the result for an OVAL Test is
        determined when using an OVAL System
        Characteristics document. An OVAL System
        Characteristics document can contain an
        optional collected_objects section. When
        the collected_objects section is present
        the following rules specify how the
        overall result for an OVAL Test is
        determined: When an
        oval-sc:collected_objects/oval-sc:object
        with an id that matches the OVAL Object id
        that is referenced by the OVAL Test is not
        found, the result for the OVAL Test must
        be "unknown". When the flag attribute of
        the corresponding
        oval-sc:collected_objects/oval-sc:object
        is "error", the result of the OVAL Test
        must be "error". When the flag attribute
```

of the corresponding oval-sc:collected_objects/oval-sc:object is "not collected", the result of the OVAL Test must be "unknown". When the flag attribute of the corresponding oval-sc:collected_objects/oval-sc:object is "not applicable", the result of the OVAL Test must be "not applicable". When the flag attribute of the corresponding oval-sc:collected_objects/oval-sc:object is "does not exist", the result of the OVAL Test is determined by examining the check_existence attribute's value and if the check_existence attribute is "none_exist" or "any_exist" the OVAL Test should evaluate to "true", for all other values of the check_existence attribute the OVAL Test should evaluate to "false". The check and state_operator attributes do not need to be considered in this condition. When the flag attribute of the corresponding oval-sc:collected_objects/oval-sc:object is "complete", the result of the OVAL Test is determined by first evaluating the check_existence attribute specified by the OVAL Test and then evaluating the check and state_operator attributes. The check attribute only needs to be considered if the result of evaluating the check_existence attribute is "true". When the flag attribute of the corresponding oval-sc:collected_objects/oval-sc:object is "incomplete", the result of the OVAL Test must be "unknown" with the following exceptions: 1) When the check_existence attribute of the OVAL Test is set to "none_exist" and the collected object has 1 or more item references with a status of "exists", a result of "false" must be reported; 2) When the check_existence attribute of the OVAL Test is set to "only_one_exists", the collected object has more than 1 item reference with a status of "exists", a result of "false" must be reported; 3) If after evaluating the check_existence attribute a non "true" result has not been determined, the check

```
          attribute must be considered as follows:
          3a) If the check attribute evaluation
          results in "false", then the OVAL Test
          result must be "false"; 3b) If the check
          attribute is set to
          "at_least_one_satisfies" and its
          evaluation results in "true", the OVAL
          Test result must be "true". When the
          collected_objects section is not present
          in the OVAL System Characteristics
          document, the evaluation engine must
          search the system characteristics for all
          Items that match the OVAL Object
          referenced by the OVAL Test. The set of
          matching OVAL Items is then evaluated
          first based on the check_existence
          attribute, then the check attribute, and
          finally the state_operator
          attribute.</xsd:documentation>
        <xsd:appinfo>
          <sch:pattern id="oval-res_testids">
            <sch:rule context="oval-res:test">
              <sch:assert
                test="@test_id = ../../oval-res:definitions//
                oval-res:criterion/@test_ref"
                  ><sch:value-of select="@test_id"/>
                - the specified test is not used in
                any definition's
                criteria</sch:assert>
            </sch:rule>
          </sch:pattern>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="message"
          type="oval:MessageType" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="tested_item"
          type="oval-res:TestedItemType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="tested_variable"
          type="oval-res:TestedVariableType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="test_id"
        type="oval:TestIDPattern" use="required"/>
      <xsd:attribute name="version"
        type="xsd:nonNegativeInteger" use="required"/>
```

```
      <xsd:attribute name="variable_instance"
        type="xsd:nonNegativeInteger" use="optional"
        default="1"/>
      <xsd:attribute name="check_existence"
        type="oval:ExistenceEnumeration"
        use="optional" default="at_least_one_exists"/>
      <xsd:attribute name="check"
        type="oval:CheckEnumeration" use="required"/>
      <xsd:attribute name="state_operator"
        type="oval:OperatorEnumeration"
        use="optional" default="AND"/>
      <xsd:attribute name="result"
        type="oval-res:ResultEnumeration"
        use="required"/>
    </xsd:complexType>
    <xsd:complexType name="TestedItemType">
      <xsd:annotation>
        <xsd:documentation>The TestedItemType
          complex type holds a reference to a system
          characteristic item that matched the
          object specified in a test. Details of the
          item can be found in the
          oval_system_characteristics section of the
          OVAL Results document by using the
          required item_id. The optional message
          element holds an error message or some
          other message that the analysis engine
          wishes to pass along. The required result
          attribute holds the result of the
          evaluation of the individual item as it
          relates to the state specified by the
          test. If the test did not include a state
          reference then the result attribute will
          be set to 'not evaluated'. Please refer to
          the description of the ResultEnumeration
          for details about the different result
          values.</xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="message"
          type="oval:MessageType" minOccurs="0"
          maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="item_id"
        type="oval:ItemIDPattern" use="required"/>
      <xsd:attribute name="result"
        type="oval-res:ResultEnumeration"
        use="required"/>
```

```
    </xsd:complexType>
    <xsd:complexType name="TestedVariableType">
      <xsd:annotation>
        <xsd:documentation>The TestedVariableType
          complex type holds the value of a variable
          used during the evaluation of a test. Of
          special importance are the values of any
          external variables used since these values
          are not captured in either the definition
          or system characteristic documents. If a
          variable is represented by a collection of
          values, then multiple elements of
          TestedVariableType, each with the same
          variable_id attribute, would exist. The
          required variable_id attribute is the
          unique id of the variable that was
          used.</xsd:documentation>
      </xsd:annotation>
      <xsd:simpleContent>
        <xsd:extension base="xsd:anySimpleType">
          <xsd:attribute name="variable_id"
            type="oval:VariableIDPattern"
            use="required"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  <!-- ======================================================== -->
  <!-- ==================  SIGNATURE  ======================== -->
  <!-- ======================================================== -->
    <!--
                  The signature element is defined by the xmldsig
                  schema.  Please refer to that documentation for
                  a description of the valid elements and types.
                  More information about the official W3C
                  Recommendation regarding XML digital signatures
                  can be found at http://www.w3.org/TR/xmldsig-core/.
          -->
  <!-- ======================================================== -->
  <!-- =================  ENUMERATIONS  ======================= -->
  <!-- ======================================================== -->
    <xsd:simpleType name="ContentEnumeration">
      <xsd:annotation>
        <xsd:documentation>The ContentEnumeration
          defines the valid values for the
          directives controlling the amount of
          expected depth found in the results
          document. Each directive specified at the
          top of an OVAL Results document defines
```

```
          how much information should be included in
          the document for each of the different
          result types. The amount of content that
          is expected with each value is defined by
          Schematron statements embedded throughout
          the OVAL Results Schema. Currently, the
          enumeration defines two values: thin and
          full. Please refer to the documentation of
          each individual value of this enumeration
          for more information about what each
          means.</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="thin">
          <xsd:annotation>
            <xsd:documentation>A value of 'thin'
              means only the minimal amount of
              information will be provided. This is
              the id associated with an evaluated
              OVAL Definition and the result of the
              evaluation. The criteria child element
              of a definition should not be present
              when providing thin results. In
              addition, system characteristic
              information for the objects used by
              the given definition should not be
              presented.</xsd:documentation>
          </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="full">
          <xsd:annotation>
            <xsd:documentation>A value of 'full'
              means that very detailed information
              will be provided allowing in-depth
              reports to be generated from the
              results. In addition to the results of
              the evaluated definition, the results
              of all extended definitions and tests
              included in the criteria as well as
              the actual information collected off
              the system must be
              presented.</xsd:documentation>
          </xsd:annotation>
        </xsd:enumeration>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="ResultEnumeration">
      <xsd:annotation>
```

```
          <xsd:documentation>The ResultEnumeration
            defines the acceptable result values for
            the DefinitionType, CriteriaType,
            CriterionType, ExtendDefinitionType,
            TestType, and TestedItemType
            constructs.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="true">
            <xsd:annotation>
              <xsd:documentation>When evaluating a
                definition or test, a result value of
                'true' means that the characteristics
                being evaluated match the information
                represented in the system
                characteristic document. When
                evaluating a tested_item, and a state
                exists, a result value of 'true'
                indicates that the item matches the
                state.</xsd:documentation>
            </xsd:annotation>
          </xsd:enumeration>
          <xsd:enumeration value="false">
            <xsd:annotation>
              <xsd:documentation>When evaluating a
                definition or test, a result value of
                'false' means that the characteristics
                being evaluated do not match the
                information represented in the system
                characteristic document. When
                evaluating a tested_item, and a state
                exists, a result value of 'false'
                indicates that the item does not match
                the state.</xsd:documentation>
            </xsd:annotation>
          </xsd:enumeration>
          <xsd:enumeration value="unknown">
            <xsd:annotation>
              <xsd:documentation>When evaluating a
                definition or test, a result value of
                'unknown' means that the
                characteristics being evaluated cannot
                be found in the system characteristic
                document (or the characteristics can
                be found but collected object flag is
                'not collected'). For example, assume
                that a definition tests a file, but
                data pertaining to that file cannot be
```

```
            found and is not recorded in the
            System Characteristics document. The
            lack of an item (in the system_data
            section) for this file in the System
            Characteristics document means that no
            attempt was made to collect
            information about the file. In this
            situation, there is no way of knowing
            what the result would be if the file
            was collected. Note that finding a
            collected_object element in the system
            characteristic document is not the
            same as finding a matching element of
            the system. When evaluating an OVAL
            Test, the lack of a matching object on
            a system (for example, file not found)
            does not cause a result of unknown
            since an test considers both the state
            of an item and its existence. In this
            case the test result would be based on
            the existence check specified by the
            check_existence attribute on the test.
            When evaluating a tested_item, and a
            state exists, a result value of
            'unknown' indicates that it could not
            be determined whether or not the item
            and state match. For example, if a
            registry_object with a hive equal to
            HKEY_LOCAL_MACHINE, a key with the
            xsi:nil attribute set to 'true', and a
            name with the xsi:nil attribute set to
            'true' was collected and compared
            against a registry_state with key
            entity equal to 'SOFTWARE', the
            tested_item result would be 'unknown'
            because an assertion of whether or not
            the item matches the state could not
            be determined since the key entity of
            the item was not
            collected.</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="error">
        <xsd:annotation>
          <xsd:documentation>When evaluating a
            definition or test, a result value of
            'error' means that the characteristics
            being evaluated exist in the system
```

```
        characteristic document but there was
        an error either collecting information
        or in performing analysis. For
        example, if there was an error
        returned by an api when trying to
        determine if an object exists on a
        system. Another example would be:
        xsi:nil might be set on an object
        entity, but then the entity is
        compared to a state entity with a
        value, thus producing an error. When
        evaluating a tested_item, and a state
        exists, a result value of 'error'
        indicates that there was either an
        error collecting the item or there was
        an error analyzing the item against
        the state. For example, a tested_item
        will receive a result value of 'error'
        if an attempt is made to compare a
        state entity against an item entity
        that has a status of
        'error'.</xsd:documentation>
    </xsd:annotation>
  </xsd:enumeration>
  <xsd:enumeration value="not evaluated">
    <xsd:annotation>
      <xsd:documentation>When evaluating a
        definition or test, a result value of
        'not evaluated' means that a choice
        was made not to evaluate the given
        definition or test. The actual result
        is not known since if evaluation had
        occurred the result could have been
        either true or false. When evaluating
        a tested_item, a result value of 'not
        evaluated' indicates that a state was
        not specified and is equivalent to an
        existence check.</xsd:documentation>
    </xsd:annotation>
  </xsd:enumeration>
  <xsd:enumeration value="not applicable">
    <xsd:annotation>
      <xsd:documentation>When evaluating a
        definition or test, a result value of
        'not applicable' means that the
        definition or test being evaluated is
        not valid on the given platform. For
        example, trying to collect Linux RPM
```

```
              information on a Windows system is not
              possible and so a result of not
              applicable is used. Another example
              would be in trying to collect RPM
              information on a linux system that
              does not have the RPM packaging system
              installed.</xsd:documentation>
          </xsd:annotation>
        </xsd:enumeration>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
```

18. Intellectual Property Considerations

DHS, on behalf of the United States, owns the registered OVAL trademarks, identifying the OVAL STANDARDS SUITE and any component part, as that suite has been provided to the IETF Trust. A "(R)" will be used in conjunction with the first use of any OVAL trademark in any document or publication in recognition of DHS's trademark ownership.

19. Acknowledgements

The authors wish to thank DHS for sponsoring the OVAL effort over the years which has made this work possible. The authors also wish to thank the original authors of this document Jonathan Baker, Matthew Hansbury, and Daniel Haynes of the MITRE Corporation as well as the OVAL Community for its assistance in contributing and reviewing the original document. The authors would also like to acknowledge Dave Waltermire of NIST for his contribution to the development of the original document.

20. IANA Considerations

This memo includes no request to IANA.

21. Security Considerations

While OVAL is just a set of data models and does not directly introduce security concerns, it does provide a mechanism by which to represent endpoint posture assessment information. This information could be extremely valuable to an attacker allowing them to learn about very sensitive information including, but, not limited to: security policies, systems on the network, criticality of systems,

software and hardware inventory, patch levels, user accounts and much more. To address this concern, all endpoint posture assessment information should be protected while in transit and at rest. Furthermore, it should only be shared with parties that are authorized to receive it.

Another possible security concern is due to the fact that content expressed as OVAL has the ability to impact how a security tool operates. For example, content may instruct a tool to collect certain information off a system or may be used to drive follow-up actions like remediation. As a result, it is important for security tools to ensure that they are obtaining OVAL content from a trusted source, that it has not been modified in transit, and that proper validation is performed in order to ensure it does not contain malicious data.

22. Change Log

22.1. -00 to -01

There are no textual changes associated with this revision. This revision simply reflects a resubmission of the document so that it remains in active status.

23. References

23.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

23.2. Informative References

[OVAL-WEBSITE] The MITRE Corporation, "The Open Vulnerability and Assessment Language", 2015, <http://ovalproject.github.io/>.

Authors' Addresses

Michael Cokus
The MITRE Corporation
903 Enterprise Parkway, Suite 200
Hampton, VA 23666
USA

Email: msc@mitre.org

Daniel Haynes
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: dhaynes@mitre.org

David Rothenberg
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: drothenberg@mitre.org

Juan Gonzalez
Department of Homeland Security
245 Murray Lane
Washington, DC 20548
USA

Email: juan.gonzalez@dhs.gov