Common Result Format (CRF™)

Specification

Version 0.3

Jon Baker
Andrew Buttner
Todd Wittbold
The MITRE Corporation

1 Introduction 3
2 Use Cases 3
3 Related Preexisting Formats 3
3.1 XCCDF Results Format 3
3.2 OVAL Results 4
4 Requirements 4
4.1 Representation 4
4.2 Simplicity 4
4.3 Extensibility 4
4.4 Asset Metadata 4
4.5 Compatibility with Preexisting Formats 5
4.6 Data Integrity 5
4.7 Additional Metadata 5
4.8 Common Asset Model Support 5
4.9 Aggregation Support 5
4.10 Data Streaming 5
4.11 Patch Naming 5
5 Detailed Design 5
5.1 Data Model 5
5.1.1 ResultSet 7
5.1.2 Generator 7
5.1.3 Assessor 7
5.1.4 Asset 8
5.1.5 AssetId 8
5.1.6 Interface 8
5.1.7 Property 8
5.1.8 PropertyNameEnum 9
5.1.9 Finding 9
5.1.10 FindingTypeEnum 10
5.1.11 ResultEnum 10
5.1.12 Check 11
5.1.13 CheckExport 11
5.1.14 CheckContentReference 11
5.1.15 CheckResultReference 11
5.2 XML Representation 12
5.2.1 XML Schema 12
5.2.2 Element Dictionary 12
5.2.3 Example XML Document 12

1 Introduction

This document describes a standardized IT asset assessment result format, Common Result Format™ (CRF™). The CRF aims to facilitate the exchange of assessment results between systems to allow for increased tool interoperability and aggregation of assessment results across large enterprises that utilize diverse technologies to detect patch levels, policy compliance, vulnerability, and asset inventory.

CRF expresses assessment results for IT assets in terms of common names and naming schemes. CRF leverages existing standardization efforts that define common names or naming schemes for vulnerabilities (CVE®), configurations (CCE™), platforms (CPE™), software weaknesses (CWE™), and patches (patch binary name). Leveraging these efforts allows meaningful assessment result data to be exchanged in a form that many tools already support or can easily develop support for. Choosing a set of standard names for each type of assessment result will the assessment results obtained from many different applications to be combined in a meaningful way. This combined set of assessments results can then easily be manipulated to allow for enterprise wide reporting.

A primary goal of CRF is to allow customers that want to utilize SCAP and SCAP Compliant tools to easily export their assessment data from preexisting non-SCAP Compliant systems to other SCAP Compliant systems. In this case detailed assessment information including checking logic and low level results may not be available in a standardized format but high level results based on the standardized names and naming schemes mentioned above can still be shared between systems.

While facilitating the sharing of information from systems that utilize proprietary checking systems for asset assessment, CRF also accommodates the exchange of standardized detailed checking logic and low level results. If detailed checking and result data from an assessment is available in XCCDF and OVAL, references to it can be included in CRF. Allowing this sort of flexibility ensures that data is only lost if it is in a non-standard format that other tools can not understand while ensuring that the format is useful for tools that have already adopted XCCDF and OVAL and want to exchange and aggregate assessment results for sets of systems.

2 Use Cases

Comment [JB1]: Coming soon

3 Related Preexisting Formats

The following related preexisting formats were considered when developing CRF.

3.1 XCCDF Results Format

The XCCDF standard includes a <TestResult> element that is used to “encapsulate the result of applying a Benchmark to one target system”. Put another way, “A TestResult object represents the results of a single application of the Benchmark to a single target platform”. This association is captured via rule ids that are a required piece of each result.

The information being captured about a particular system evaluation is very similar to the desired goals of CRF. XCCDF Results is simple in nature and does not contain a lot of ‘extra’ information. The basic format starts by defining a target (the asset being reported on) and then reporting the true/false result of each rule that was evaluated. Unfortunately, XCCDF Results does not support reporting on items other than XCCDF Rules.

Using the XCCDF Results format for result information that was not obtained through the evaluation of an existing Benchmark goes against the intent of the format. XCCDF Results are associated with an existing Benchmark through the rule id. In order to use XCCDF Results for reporting results based on individual

CVEs or CCEs, the Rule id would have to be replaced by a CVE/CCE id. Although simple in practice, this change would not be understood by a tool that has been developed to understand the standard. The tool would be expecting a XCCDF Rule id and would not be able to distinguish the change to some other id format.

CRF will allow tools that produce XCCDF Results to easily produce CRF documents as long as the XCCDF Results contain references to the defined set of standard names and naming schemes. CRF will also allow references to XCCDF Results documents to be included if they are available.

3.2 OVAL Results

OVAL defines a format for expressing authenticated asset assessment results based on OVAL Definition evaluations, OVAL Results. Data expressed in CRF can be derived from an OVAL Results document as long as the definitions in the OVAL Results document are based on the set of defined standard names and naming schemes. However, because OVAL Results are intended to be based on OVAL Definition evaluations, the format does not meet the goals of CRF. CRF aims to accommodate results obtained from many other methods including unauthenticated assessments of an asset.

CRF will allow tools that produce OVAL Results to easily produce CRF documents as long as the OVAL Results contain references to the defined set of standard names and naming schemes. CRF will also allow references to OVAL Results documents to be included if they are available.

4 Requirements

The following requirements have guided the development of CRF.

4.1 Representation

A detailed UML data model for CRF is included in this document. An XML Schema based on the data model is also included. An XML representation has been selected because it integrates nicely with SCAP and is a preferred mechanism for data exchange.

4.2 Simplicity

CRF attempts to err on the side of simplicity. In order to accommodate preexisting tools that contain large amounts of useful data collected for differing purposes, unneeded complexity has been avoided whenever possible to ensure that CRF is simple to use as either a producer or a consumer of data in the format.

4.3 Extensibility

CRF attempts to be extensible and accommodate the various types of data that may be collected by different tools for varying purposes. The format utilizes a minimal common set of assessment information and allows additional information to be included at the discretion of content producers.

4.4 Asset Metadata

For asset identification CRF must require only a minimal amount of metadata to uniquely identify an IT asset. CRF must also define an additional set of optional asset metadata to allow tools with additional asset information to encode it if desired. Additionally, unaccounted for asset metadata must be supported without a change to the format. Requiring only a minimal set of metadata to uniquely identify an asset allows for the greatest number of applications to produce CRF documents.

CRF must also encapsulate metadata related to the date and time an IT asset was assessed. Assessment results for a given asset might be encoded in a single CRF document many times. In this case each occurrence of the single asset would represent either assessment results generated by different applications, or results generated by the same application at different times, or both.

4.5 Compatibility with Preexisting Formats

CRF defines a set of attributes for asset identification that allow for simple translation from the asset identification structures defined in OVAL and XCCDF. CRF also attempts to follow existing structures for detailed check result representations when possible. When used in conjunction with other SCAP standards this should reduce the time and cost to adopt CRF. CRF utilizes a structure for representing detailed check information and results that is complementary to the structures defined in the XCCDF specification. Tools that already utilize XCCDF should be able to easily insert detailed check information about a particular finding.

4.6 Data Integrity

CRF allows content producers to digitally sign data using XML Signatures to support data integrity checking.

4.7 Additional Metadata

CRF defines a set of metadata related to the creation of the document, including the software that produced the document and timestamps related to its creation.

4.8 Common Asset Identification Model Support

A common asset identification model should be developed that can be shared across all related standards. When this model is developed CRF will leverage it for asset identification.

4.9 Aggregation Support

CRF will allow for a given asset to be reported on 1 or many times in a single instance document. This capability will allow various tools to report on overlapping sets of assets or a single tool to report on a single asset several different times.

4.10 Data Streaming

It is expected that a single CRF document could be very large. CRF must be designed to support streaming processing of the data it encodes. To support this requirement any references between objects in a CRF document should be defined to always refer up to previously defined objects. This will ensure that data a given object refers to will be parsed prior to the object itself.

4.11 Patch Naming

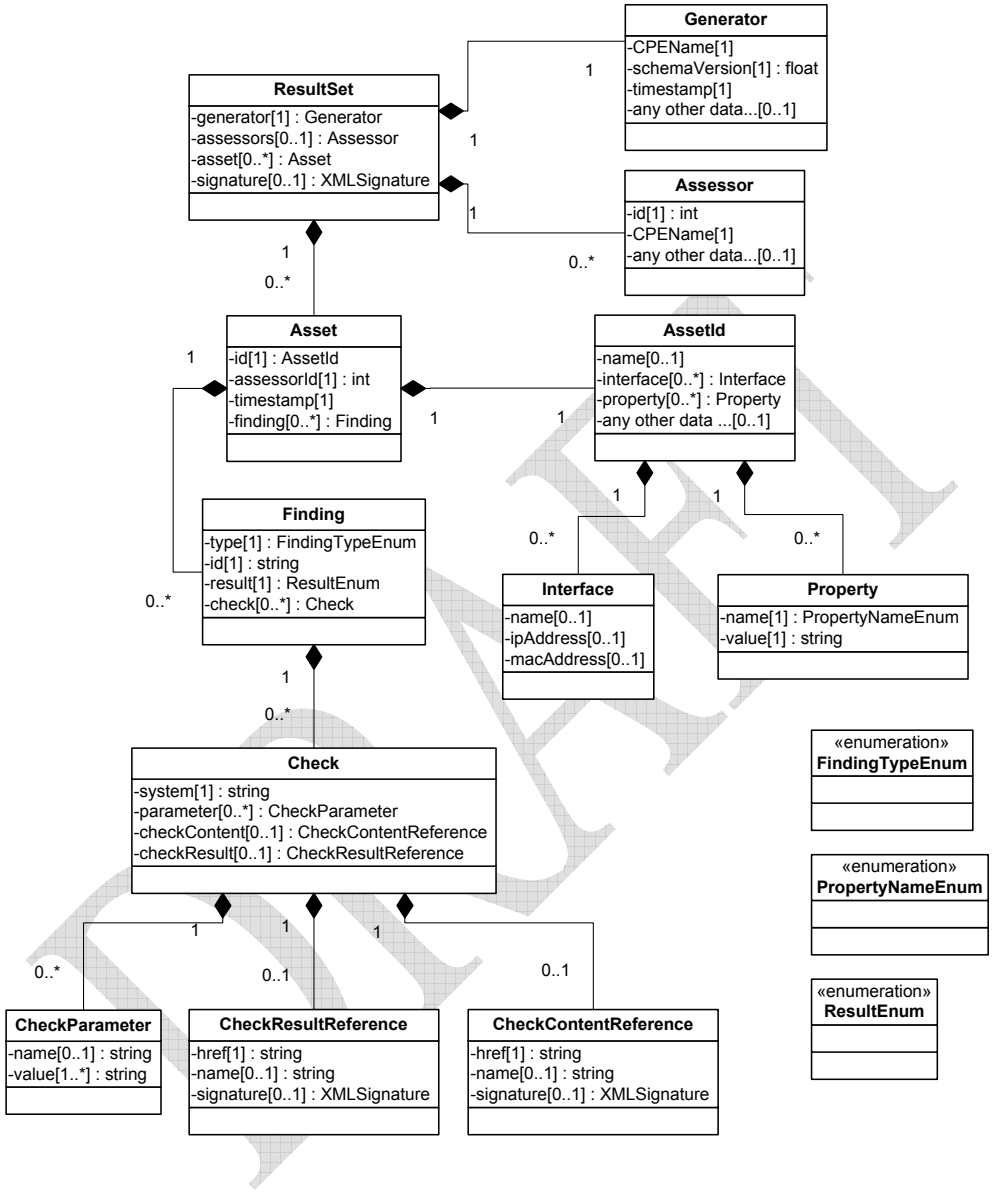
A standardized patch naming schema does not currently exist. CRF must support exchanging patch checking information for an asset. Enable to do this a common naming approach must be defined. CRF will uniquely identify patches based on their binary name. So for example, a given Microsoft security bulletin might have several patch binaries associated with it. Each of those patch binaries has a unique name. CRF will utilize this unique name to differentiate these three patches. A survey of major operating systems has shown that differentiating patches based on binary name will work for most cases.

5 Detailed Design

This section provides a detailed design description for CRF.

5.1 Data Model

The following UML diagram describes CRF. Each object in the diagram is described in detail.



5.1.1 ResultSet

The ResultSet class is a top level class that simply contains a set of 0 or more Assets, 0 or more Assessors, a Generator, and optionally a Signature. The Signature represents an XML Signature over the entire ResultSet. Each attribute of the ResultSet class is described below:

Attribute	Type	Multiplicity	Description
generator	Generator	1	Identifies the Tool that compiled the ResultSet.
assessor	Assessor	0 - *	The set of Assessors that assessed one or more of the Assets described in the ResultSet.
asset	Asset	0 - *	The set of Assets being reported on in the ResultSet
signature	XMLSignature	0 - 1	An optional XmlSignature over the entire ResultSet.

5.1.2 Generator

The Generator class represents information that identifies the system or application that compiled the ResultSet and when the ResultSet was compiled. The Generator class may be extended to accommodate any additional metadata about the Generator of the ResultSet. This class is similar in intent to the GeneratorType defined the oval-common-schema. Each attribute of the Generator class is described below:

Attribute	Type	Multiplicity	Description
CPEName	uri	1	The CPE name of the tool that compiled the ResultSet. See http://cpe.mitre.org for the definition of a CPEName.
timestamp	dateTime	1	The timestamp indicates the date and time when the ResultSet was compiled.
any	any	0 – n	A tool may add additional metadata into the Generator if needed. This data is not validated and must be defined in some namespace other than the CRF namespace.

5.1.3 Assessor

The Assessor class represents information that identifies the system or application that was used to assess one or more of the Assets in the ResultSet. The Assessor class may be extended to accommodate any additional metadata about the Assessor. Each attribute of the Assessor class is described below:

Attribute	Type	Multiplicity	Description
id	int	1	The id is an integer that uniquely identifies one Assessor in the ResultSet and serves as a simple mechanism to allow references to Assessors from Assets in the ResultSet.
CPEName	uri	1	The CPE name of the Assessor. See http://cpe.mitre.org for the definition of a CPEName.
any	any	0 – n	A tool may add additional metadata into the Assessor if needed. This data is not validated and must be defined in some namespace other than the CRF namespace.

5.1.4 Asset

The Asset class represents an IT asset and a set of findings for that asset. An Asset may be included in the ResultSet any number of times. This allows an Asset to be assessed by more than one Assessor or at different times by the same Assessor. Each attribute of the Asset class is described below:

Attribute	Type	Multiplicity	Description
id	AssetId	1	Identifies the asset based on the set of information that the Assessor was able to gather for the Asset.
assessorId	int	1	A reference to the id of the Assessor that generated the result information for the Asset.
timestamp	dateTime	1	The date and time that the result information for the asset was generated.
finding	Finding	0 – n	A set of one or more findings for the Asset.

5.1.5 AssetId

The AssetId class represents a minimum set of metadata needed to uniquely identify an IT asset. Additionally a set of defined properties may also be associated with an Asset. The AssetId class may be extended to accommodate any additional metadata about the Asset. A valid AssetId must specify a name and at least one interface. Each attribute of the AssetId class is described below:

Attribute	Type	Multiplicity	Description
name	String	1	The fully qualified domain name of the Asset.
interface	Interface	1 - n	The set of interfaces found on the Asset. At least one Interface must be identified.
property	Property	0 – n	An optional set of Properties of the Asset.
any	any	0 - n	An Assessor may add additional metadata about an Asset as needed. This data is not validated and must be defined in some namespace other than the CRF namespace.

5.1.6 Interface

The Interface class represents a network interface on an IT asset. An interface object must define at least an IP address or MAC address. Each attribute of the Interface class is described below:

Attribute	Type	Multiplicity	Description
name	string	0 - 1	The name of the interface.
ipAddress	string	0 - 1	The ip address associated with the interface.
macAddress	string	0 - 1	The MAC address of the interface.

5.1.7 Property

The Property class represents a name value pairing of asset identification metadata. The set of allowable Property names is constrained by the PropertyNameEnum datatype. Each attribute of the Property class is described below:

Attribute	Type	Multiplicity	Description
name	PropertyNameEnum	1	The name of the property.
value	string	1	The value of the property

5.1.8 PropertyNameEnum

Comment [JB2]: Currently defined in the crf spec, but need to be moved out to an external source. This set will change overtime. This list is based on a list sent from Joe Wolfkiel.

The PropertyNameEnum datatype defines the set of allowable property names. Future versions of this specification may separate the allowed set of property names into either an external document or a separate section of this document The set of allowed property names is defined below:

Name	Description
urn:crf:asset:identifier:mac	should be sent as a pair with the ip address to ensure uniqueness
urn:crf:asset:identifier:ipv4	internet protocol version 4 address
urn:crf:asset:identifier:ipv6	internet protocol version 6 address
urn:crf:asset:identifier:fqdn	fully qualified domain name
urn:crf:asset:identifier:ein	equipment identification number
urn:crf:asset:identifier:pki	any device pki certificates as a text field
urn:crf:asset:identifier:guid	globally unique identifier
urn:crf:asset:identifier:ldap	LDAP directory string
urn:crf:asset:identifier:active_directory	Active Directory realm
urn:crf:asset:identifier:host_name	host name if assigned to an asset
urn:crf:asset:environmental_information:owning_organization	organization that tracks the asset on their ADPE inventory
urn:crf:asset:environmental_information:current_region	geographic region {e.g. COCOM AOR} where the asset is located
urn:crf:asset:environmental_information:administration_unit	name of the organization that does system administration for the asset
urn:crf:asset:environmental_information:cnd_service_provider	name of the CERT equivalent that does incident response for an asset
urn:crf:asset:environmental_information:administration_poc:title	Title {e.g. Mr, Ms, Col} of the system administrator for an asset
urn:crf:asset:environmental_information:administration_poc:e-mail	e-mail of the system administrator for an asset
urn:crf:asset:environmental_information:administration_poc:first_name	first name of the system administrator for an asset
urn:crf:asset:environmental_information:administration_poc:last_name	Last name of the system administrator for an asset

5.1.9 Finding

The Finding class represents a finding related to an asset. Findings have a type based on the FindingTypeEnumeration. The type of a Finding identifies the common name or naming scheme that the Finding is based on. The id of the Finding identifies a single item with in the common name or naming scheme identified by the type. The result specifies the outcome of checking the Asset for the identified item. The Finding class also allows for one or more Checks to be associated with a Finding object. Each attribute of the Finding class is described below:

Attribute	Type	Multiplicity	Description
type	FindingTypeEnum	1	The type of the item being reported on. The allowable set of finding types is defined by the FindingTypeEnum.
id	string	1	The id of the item being reported on. The structure of

			the id is based on the type of finding.
result	ResultEnum	1	The result of the item being reported on. The allowable result values are defined by the ResultEnum.
check	Check	0 - n	The optional set of Checks to be associated with a Finding.

The following table describes the meaning of the true and false ResultEnum values based in the FindingTypeEnum value:

Finding Type	result = true	result = false
CCE	The identified configuration item passed its check based on the parameters in use in the environment on the asset.	The identified configuration item failed its check based on the parameters in use in the environment on the asset.
CPE	The identified application, operating system, or hardware was detected on the asset.	The identified application, operating system, or hardware was detected on the asset.
CVE	The identified vulnerability was detected on the asset.	The identified vulnerability was not detected on the asset.
CWE	The identified weakness was detected on the asset.	The identified weakness was not detected on the asset.
Patch	The identified patch should be installed on the asset.	The identified patch should not be installed on the asset.

5.1.10 FindingTypeEnum

The FindingTypeEnum defines the set of supported Finding types. This set of Finding types is expected to change over time. Future versions of this specification may separate the allowed set of Finding types into either an external document or a separate section of this document. Each type is listed and described below:

Name	Description
CCE	CRF uses CCE to identify configuration items. See http://cce.mitre.org for more information about CCE and the structure of a CCE identifier.
CPE	See http://cpe.mitre.org for more information about CPE and the structure of a CPE Name.
CVE	CRF uses CVE to name vulnerabilities. A value of CVE indicates that that type of finding is based on a CVE name. See http://cve.mitre.org for more information about CVE and the structure of a CVE name.
CWE	See http://cwe.mnitre.org for more information about CWE and the structure of a CWE identifier.
Patch	See the “Patch Naming” section of this document for more information about the structure of a patch names.

5.1.11 ResultEnum

The ResultEnum defines the set of possible result values. Each value is listed and defined below:

Name	Description
true	The meaning of a ‘true’ result varies depending on the type of Finding. See the documentation of the Finding class for a table that defines the meaning of a ‘true’ result for each type of Finding.
false	The meaning of a ‘false’ result varies depending on the type of Finding. See the documentation of the Finding class for a table that defines the meaning of

	a ‘false’ result for each type of Finding.
error	A result value of 'error' means that the assessor encountered an error while assessing the asset.
unknown	A result value of 'unknown' means that the Assessor was unable to determine a ‘true’ or ‘false’ result.
not evaluated	A choice was made not to check the asset for the identified item. The actual result is in essence unknown since if evaluation had occurred it could have been either true or false.
not applicable	The Assessor determined that the specified item is not valid on the Asset.

5.1.12 Check

The Check class represents information about a check, a reference to the check, any parameters passed to the check, and the results of the check or references to the results of the check used by an Assessor to determine a result for a Finding. Each attribute of the Check class is described below:

Attribute	Type	Multiplicity	Description
system	string	1	The system attribute identifies the checking system used. Allowable checking systems are either OVAL or XCCDF.
checkExport	CheckExport	0 - n	The checkExport attribute allows Assessors to include parameters used by the Check.
checkContent	AbsCheckContent	0 - 1	The checkContent attribute allows Assessors to include inline or by reference the actual check used.
checkResult	AbsCheckResult	0 - 1	The checkResult attribute allows Assessors to include inline or by reference the actual detailed check results.

5.1.13 CheckExport

The CheckExport class represents a parameter that was used by a Check. Each attribute of the CheckContentReference class is described below:

Attribute	Type	Multiplicity	Description
name	string	1	Name of the parameter exported to the check.
value	string	1 - n	The set of values provided for the named parameter.

5.1.14 CheckContentReference

The CheckContentReference class allows detailed check content to be associated with a check by reference. Each attribute of the CheckContentReference class is described below:

Attribute	Type	Multiplicity	Description
href	uri	1	A pointer to the document containing the check.
name	string	0 - 1	The name of the check in the referenced document. If not specified then the check is the entire referenced document.
signature	XMlSignature	0 - 1	Optional Xml Signature over the contents of the external document.

5.1.15 CheckResultReference

The CheckResultReference class represents a reference to an external document containing detailed result information for a Check.

Attribute	Type	Multiplicity	Description

href	uri	1	A pointer to the document containing the check results.
name	string	0 - 1	The name of the check in the referenced document. If not specified then the check is the entire referenced document.
signature	XMlSignature	0 - 1	Optional Xml Signature over the contents of the external document.

5.2 XML Representation

5.2.1 XML Schema

The XML Schema below defines the structure of CRF's XML representation.

Comment [JB3]: How will scheamtron be used if at all?

5.2.2 Element Dictionary

5.2.3 Example XML Document

The xml below is provided as a simple point of reference for those that are more comfortable looking at an xml document as an example. Once a schema and element dictionary is added to this document this example will be updated to comply with the schema.

The document below was compiled by some tool, "`cpe:/a:some_vendor0:some_product`" and contains assessment results from two other tools that assessed two different assets.

```
<?xml version="1.0" encoding="UTF-8"?>
<CommonResultSet
    xmlns="http://crf.mitre.org/crf-0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://crf.mitre.org/crf-0.xsd crf.xsd"
    xmlns:crf="http://crf.mitre.org/crf-0.xsd">
    <Generator>
        <CPEName>cpe:/a:some_vendor0:some_porduct</CPEName>
        <schema_version>0.2</schema_version>
        <timestamp>2007-09-10T15:00:00.000</timestamp>
        <!-- any other structured data -->
    </Generator>
    <Assessors>
        <Assessor id="1">
            <CPEName>cpe:/a:some_vendor1:some_porduct</CPEName>
            <!-- any other structured data -->
        </Assessor>
        <Assessor id="2">
            <CPEName>cpe:/a:some_vendor2:some_porduct</CPEName>
            <!-- any other structured data not definied in the specification -->
        </Assessor>
    </Assessors>
    <Assets>
        <Asset assessorId="1" timestamp="2007-09-10T13:00:00.000">
            <Id>
                <name>asystem.example.com</name>
                <interfaces>
                    <!-- one interface is required. -->
                    <interface>
                        <name>Broadcom NetXtreme 57xx Gigabit Controller #2</name>
                        <ip_address>192.0.0.1</ip_address>
                        <mac_address>00-AA-C5-BC-1A-A0</mac_address>
                    </interface>
                    <!-- additional interfaces are supported -->
                </interfaces>
                <properties>
                    <!-- optional list of defined properties -->
                    <property name="urn:crf:asset:identifier:ipv4">192.0.1.1</property>
                </properties>
```

```
            <!-- any other structured data not defined in the specification -->
        </Id>
        <Findings>
            <Finding type="CVE" id="CVE-2007-1234" result="true">
                <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                    <contentReference href="oval.xml" name="oval:com.example:def:1"/>
                    <resultReference href="oval-results.xml" name="oval:com.example:def:1"/>
                </check>
            </Finding>
            <Finding type="CCE" id="CCE-123" result="true">
                <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                    <parameter name="oval:com.example:var:1">true</parameter>
                    <parameter export-name="oval:com.example:var:2">8</parameter>
                    <contentReference href="oval.xml" name="oval:com.example:def:2"/>
                    <resultReference href="oval-results.xml" name="oval:com.example:def:2"/>
                </check>
            </Finding>
            <Finding type="CPE" id="cpe:/o:microsoft:windows:xp" result="true">
                <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                    <contentReference href="oval.xml" name="oval:com.example:def:4"/>
                    <resultReference href="oval-results.xml" name="oval:com.example:def:4"/>
                </check>
            </Finding>
            <Finding type="CPE" id="cpe:/o:microsoft:windows:2003" result="false">
                <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                    <contentReference href="oval.xml" name="oval:com.example:def:5"/>
                    <resultReference href="oval-results.xml" name="oval:com.example:def:5"/>
                </check>
            </Finding>
        </Findings>
    </Asset>
    <Asset assessorId="2" timestamp="2007-09-10T14:00:00.000">
        <Id>
            <name>mysystem.example.com</name>
            <interfaces>
                <!-- one interface is required. -->
                <interface>
                    <name>Broadcom NetXtreme 57xx Gigabit Controller #2</name>
                    <ip_address>192.0.0.2</ip_address>
                    <mac_address>00-AA-C5-BC-1A-A1</mac_address>
                </interface>
                <!-- additional interfaces are supported -->
            </interfaces>
            <properties>
                <!-- optional list of defined properties -->
                <property name="urn:crf:asset:identifier:ipv4">192.0.1.2</property>
            </properties>
            <!-- any other structured data not defined in the specification -->
        </Id>
        <Findings>
            <Finding type="CVE" id="CVE-2007-1234" result="false"/>
            <Finding type="CCE" id="CCE-123" result="true"/>
            <Finding type="CPE" id="cpe:/o:redhat:linux" result="true"/>
            <Finding type="CPE" id="cpe:/a:mozilla:firefox" result="true"/>
        </Findings>
    </Asset>
  </Assets>
</CommonResultSet>
```