

INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: March 28, 2013

Mark Davidson
The MITRE Corporation
David Oliva
September 24, 2012

Asset Summary Reporting
draft-davidson-sacm-asr-00

Abstract

This specification defines the Asset Summary Reporting (ASR), a data model for expressing the data exchange format of summary information relative to one or more metrics. ASR reduces the bandwidth requirement to report information about assets in the aggregate since it allows for reporting aggregates relative to metrics, as opposed to reporting data about each individual asset, which can lead to a bloated data exchange. ASR is vendor neutral and leverages widely adopted, open specifications; it is flexible, and suited for a wide variety of reporting applications.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	4
1.1	Purpose and Scope	4
1.2	Audience	4
1.3	Document Structure	5
1.4	Document Conventions	5
2	Terms and Abbreviations	6
2.1	Terms	6
2.2	Acronyms	7
3	Relationships to Existing Standards and Specifications	8
4	Conformance	9
4.1	Product Conformance	9
4.2	Content Conformance	9
5	ASR Data Model Overview and Key Concepts	9
5.1	Data Model Overview	9
5.1.1	Summary Report	10
5.1.2	Record Set	10
5.1.3	Record	10
5.1.4	Data Source	10
5.1.5	Metadata	10
5.2	Key Concepts	10
5.2.1	Record Attributes	11
5.2.2	Record Set Type	11
5.2.3	Paging	11
5.2.4	Referencing Assets	12
6	ASR Data Model Description	12
6.1	XML Data Model Introduction	12
6.2	XML Data Model Requirements	15
7	Defining a Record Set Type	18
8	IANA Considerations	19
9	Security Considerations	19
10	References	20
10.1	Normative References	20

10.2 Informative References	21
Appendix A Acknowledgements	21
Appendix B Use Cases	21
A.1 Continuous Monitoring	21
A.2 Security Automation	21
Appendix C Integration with ARF	22
Appendix D Record Set Example	23
Appendix E Sample Record Set Type Definitions	25
F.1 SCAP Attributes	29
F.3 Statistical Attributes	29
F.4 Other Attributes	29
E.2 Finding Attributes	29
Authors' Addresses	29

1 Introduction

The Asset Summary Reporting (ASR) format is a data model to express the transport format of summary information about one or more sets of assets. The data model facilitates the interchange of aggregated asset information throughout and between organizations. ASR is vendor neutral and leverages widely adopted, open specifications; it is flexible, and suited for a wide variety of reporting applications. The primary goal of the ASR format is to describe summary information about one or more arbitrarily large and complex asset-related data sets in a standardized manner. Second, ASR seeks to allow content producers the ability to choose an appropriate level of detail depending on their needs and data set size requirements. Finally, ASR seeks to reduce the complexity of producing and consuming summary result documents. For the purposes of this specification, an asset is considered to be anything that has value to an organization. Computing devices are one form of asset that many organizations track. Additional examples are networks, people, and organizations. This specification, however, does not limit asset summary reporting to those examples; information about any set of assets may be summarized. While this specification was developed to support the immediate needs of the security automation and the continuous monitoring communities, it is expected that this specification will be valuable to any process where producing or consuming summary data is desired.

1.1 Purpose and Scope

The purpose of this report is to define the ASR specification. The report gives an introduction to ASR version 1.0, defines ASR's data model, and documents the conformance requirements to comply with ASR. Other versions of ASR and the associated component specifications, including emerging specifications and future versions, are not addressed here. Future versions of ASR will be defined in distinct revisions of this report, each clearly labeled with a revision number and the appropriate ASR version number. This report does not describe the queries, instructions, methods, processes, or data required to produce an ASR document. This report does not describe how to transform any specific data model or data set into an ASR document. This report normatively describes only the ASR format. The appendices contain additional information about how to use ASR.

1.2 Audience

The intended audience for this specification is product vendors who are developing applications that either produce or consume aggregated asset information, particularly those that will interoperate with other producers or consumers of aggregated asset information.

1.3 Document Structure

The remainder of this document is organized into the following major sections:

Section 2 defines the terms used within this specification and provides a list of common abbreviations.

Section 3 describes how this specification relates to other standards and specifications.

Section 4 defines the conformance requirements for ASR.

Section 5 provides an overview of the ASR data model constructs and key concepts.

Section 6 documents the ASR data model.

Section 7 specifies the requirements for defining a record set type.

Appendix A describes use cases for ASR.

Appendix B indicates how to integrate ASR with the Asset Reporting Format (ARF).

Appendix C provides some ASR examples.

Appendix D provides examples of record set type definitions.

Appendix E lists pre-defined record attributes.

Appendix F documents normative references.

1.4 Document Conventions

Throughout this specification, when referencing a specification listed in Appendix F, the name will be written between brackets, such as [XSD].

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].

XML elements [XML] are referred to using qualified names when they are not in the ASR namespace. Elements with no prefix can be assumed to be in the ASR namespace, unless otherwise noted. A qualified name associates a named element with a namespace. The namespace identifies

the specific XML schema that defines (and consequently may be used to validate) the syntax of the element instance. A qualified name declares this schema to element association using the format 'prefix:element-name'. The association of prefix to namespace is defined in the metadata of an XML document and varies from document to document. In this specification, the conventional mappings listed in Table 1-1 are used.

Table 1-1: Conventional XML Mappings

	Mappings Prefix	Namespace URI	Schema
	ai	http://scap.nist.gov/schema/asset-identification/1.1	Asset Identification 1.1
RF	arf-rel	http://scap.nist.gov/specifications/arf/vocabulary/relationships/1.0#	A 1.1 Relationships
	asr	http://scap.nist.gov/schema/asset-summary-reporting/1.0	ASR 1.0
	asr-attr	http://scap.nist.gov/schema/asset-summary-reporting/1.0/attr	ASR 1.0 Common Attributes

2 Terms and Abbreviations

This section defines a set of common terms and abbreviations used within this specification.

2.1 Terms

Asset: Anything that has value to an organization, including, but not limited to, another organization, person, computing device, information technology (IT) system, IT network, IT circuit, software (both an installed instance and a physical instance), virtual computing platform (common in cloud and virtualized computing), and related hardware (e.g., locks, cabinets, keyboards).

Asset Identification: The attributes and methods necessary for uniquely identifying a given asset. A full explanation of asset identification is provided in [Asset Identification].

Data Source: Represents a source of data that could be used to build a summary report.

Population: A defined set of assets from which reporting is based.

Record Set: A grouping of related data about a topic, organized into

records and data elements.

Record Set Type: A description of a record set that defines requirements for the record set, and the semantics of the data fields.

Summary Report: A collection of related record sets into a coherent report, as defined by the report creator. A summary report may be represented in multiple pages, which would be manifested as multiple XML documents.

2.2 Acronyms

ARF - Asset Reporting Format

ASR - Asset Summary Reporting Format

CCE - Common Configuration Enumeration

CCSS - Common Configuration Scoring System

CISO - Chief Information Security Officer

CPE - Common Platform Enumeration

CPU - Central Processing Unit

CVE - Common Vulnerabilities and Exposures

CVSS - Common Vulnerability Scoring System

CWE - Common Weakness Enumeration

CWSS - Common Weakness Scoring System

FIPS - Federal Information Processing Standard

IETF - Internet Engineering Task Force

IR - Interagency Report

ISCM - Information Security Continuous Monitoring

IT - Information Technology

ITL - Information Technology Laboratory

NIST - National Institute of Standards and Technology

OCIL - Open Checklist Interactive Language

OS - Operating System

OVAL - Open Vulnerability and Assessment Language

RACI - Responsible, Accountable, Consult, Inform (Responsibility Matrix)

RFC - Request for Comment

SCAP - Security Content Automation Protocol

SIEM - Security Information and Event Management

SP - Special Publication

SWID - Software Identification

URI - Universal Resource Identifier

USGCB - United States Government Configuration Baseline

W3C - World Wide Web Consortium

XCCDF - Extensible Configuration Checklist Description Format

XML - Extensible Markup Language

XSD - XML Schema

XSLT - Extensible Stylesheet Language Transformations

3 Relationships to Existing Standards and Specifications

ASR's relationships to other selected specifications are described below.

1. Asset Identification - ASR leverages [Asset Identification] to identify assets for the ASR document. Asset Identification is a useful component of ASR, as it enables the correlation and fusion of various ASR documents and ASR content. ASR provides a place to optionally list assets defined using [Asset Identification].

2. Asset Reporting Format (ARF) - ASR may be used as a payload for an Asset Reporting Format (ARF) document [ARF]. ASR is not required to be packaged as an ARF payload; however, Appendix B provides guidance on using ASR in an ARF report.

4 Conformance

Developers and organizations may want to build products in conformance with ASR to foster consistency and interoperability of those products. Users of those products, and consumers of the content generated by those products, would then know the format of the data that the product will produce and can consume. In addition, products that conform to this specification will be better able to interoperate and exchange reporting information with other products that conform to ASR. Products may want to claim conformance with this specification to advertise their interoperability with other ASR compliant tools, as well as to meet requirements set by other specifications or organizations. The following sections define the criteria for content and products to claim conformance with this specification.

- 4.1 Product Conformance There are two types of ASR products: report producers and report consumers. Report producers are products that generate ASR documents, while report consumers are products that accept an existing ASR document and process it. Products claiming conformance with this specification SHALL adhere to the following requirements: 1. For report producers, generate well-formed content as defined in Section 4.2. 2. For report consumers, consume and process well-formed content as defined in Section 4.2. 3. Make an explicit claim of conformance to this specification in any documentation provided to end users.
- 4.2 Content Conformance In order for an ASR report to be considered in compliance with this specification, the report MUST adhere to the following requirements: 1. The ASR report SHALL conform to all of the normative guidance provided in Section 6. 2. Each record set within an ASR report SHALL conform to the requirements set by its declared record-set-type, as described in Section 7.

5 ASR Data Model Overview and Key Concepts

This section provides an overview of the ASR data model structure and design philosophy. The data model defines a format for representing one or more collections of data. A collection of data about assets is called a record set. At its simplest, an ASR report is a collection of record sets. The following sections introduce the key concepts of the ASR data model.

5.1 Data Model Overview

The following sections provide a high level overview of the main data model constructs.

5.1.1 Summary Report

An ASR document (i.e., a "summary report") is composed of one or more record sets. A summary report is the highest level notion in the ASR data model. One or more XML documents may compose a summary report, so the concept of a summary report is not confined to the physical boundaries of an XML document.

5.1.2 Record Set

A record set is a collection of data organized into individual records. A record set optionally provides a reference to information about the source of the data for the records. The context of a record set is communicated by declaring a "type", known as a record set type. Section 5.2.2 provides more details regarding the record set type.

5.1.3 Record

The record construct is the primary mechanism for conveying data in a summary report. All record data is contained in the attributes of the record construct. The properties of a record are defined by the record set type. The record construct may have any number of properties, as permitted by the record set type. In general, properties are expected to be qualified XML attributes as described in Section 5.2.1.

5.1.4 Data Source

In addition to capturing a collection of records, a record set may capture information about the origin of the data in its records. Data source information is captured separately from the record set, but each record set may reference a data source. The same data source may be referenced by multiple record sets if that is appropriate for the summary report.

5.1.5 Metadata

A summary report may have metadata about its creation. The metadata should include the name of the tool or person that generated the report, the time the report was generated, and any other pertinent information.

5.2 Key Concepts

The following sections provide an overview of the key concepts in composing a summary report.

5.2.1 Record Attributes

Record attributes are the core construct for construing information in ASR. A record set type defines the allowable and required attributes on a record in a particular record set. The attributes must be namespace qualified in order to give context as to the meaning of the attribute. In XML, attributes that are not namespace qualified belong to the "no namespace" realm, which does not give context to the meaning of the attribute. The namespace and local-name must be defined in the corresponding record set type.

5.2.2 Record Set Type

The concept of a record set is generic in nature, and it is based on the expectation that a report creator must define a record set type or adopt an existing record set type. A record set type is a definition of a record set. The record set type defines all properties of a record set. In object-oriented programming, a class is the functional equivalent of a record set type, and an object is the functional equivalent of a record set instance. The record set type describes the attributes that "SHOULD", "MUST", and "MAY" be included in a record in that record set. It also describes the semantics of each attribute. Section 7 gives specific requirements for defining a record set type. While a record set type must be defined, the format of the record set type is not strictly defined. A record set type definition is an agreement between producer and consumer. It is anticipated that record set type definitions may take the form of prose, XML, spoken word, or any other form of communication capable of conveying the requirements of a record set type. The authors of this specification have provided a data model and XML schema for record-set-type-definitions that may be used for this purpose. The record-settype-definition data model is covered in Section 7, and that section also provides a pointer to the recordset-type-definition XML schema.

5.2.3 Paging

Since a summary report can grow too large for available resources (e.g., network bandwidth, memory, CPU), a summary report may be divided into multiple pages. A paged summary report means that a single summary report is represented as two or more separate XML documents. This allows report creators the flexibility to reduce the resources required to produce and exchange a single large summary report by breaking it up into many smaller reports. Paging exists to support use cases where the amount of data contained in an ASR exceeds reasonable computing thresholds. Paging can be used to send multiple, smaller segments of information instead of one large block of information. Each page of an ASR should be consumable without the

other pages when possible. All paging information is contained in the root element of each XML document.

5.2.4 Referencing Assets

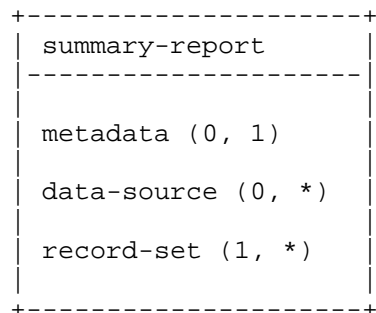
A record may be allowed or required to capture an asset list as a child. If an asset list is captured, there must be a count attribute in the record that is identified as the "count for the asset list", as defined in Section 7. That attribute must hold a value equal to the number of assets listed within the record. The "count for the asset list" attribute represents the count of assets that meet certain criteria. The criteria are specified in the description of the count attribute.

6 ASR Data Model Description

This section describes the requirements for the ASR data model manifested as Extensible Markup Language (XML). Section 6.1 provides a conceptual overview of the data model, while Section 6.2 examines the actual XML data model in detail.

6.1 XML Data Model Introduction

Figure 6-1 provides a logical view of a sample summary report spread across two pages. For brevity some attributes have been omitted. In the diagram, each record set claims a type and a data source description. Those items give additional context to understand the meaning of the data captured in the record set.



The summary-report element is the root element of the ASR data model; it contains identification and paging information for an ASR document. One ASR document may be paged into multiple XML documents as desired, but each record set **MUST** be completely represented on one page of a summary report (i.e., a record set may not span multiple

pages). Summary-report has three attributes: report-id, page-number, and last-page; it also has three children: metadata, record-set, and data-source.

The metadata element is a child of the summary-report element. Metadata contains information related to the generation of an ASR document. Metadata has two attributes: generator-name and timestamp. Metadata does not have any defined children, but there is an extension point where any arbitrary XML is allowed.

The record-set element is a child of the summary-report construct. A record-set contains summary data, and may appear multiple times in the same summary-report. Record-set has four attributes: id, data-source-ref, record-set-type, and comment. Data-source-ref is a reference to the data-source element that represents the source of the information used in the record-set, record-set-type indicates the type of report being represented, and the comment field allows a text comment, if desired. Record-set has one child: record.

The record element is a child of the record-set construct, and contains any number of attributes; record may appear multiple times in the same record-set. Record has two children: asset-references and identifier-list; only one of the two may be present for a single report. Both children provide methods to identify and list the assets that each record describes.

The identifier-list element is used to enumerate assets that relate to data in the record. In the definition for the record set type, an attribute that contains a count may be associated with this list. When that occurs, the identification information captured in this element is known to enumerate the list of assets that make up the associated count. Assets are enumerated using this element through a unique identification scheme defined by capturing the URI for the schema. Subsequently, each id provided as a child to this element is understood within the context of the identifier scheme specified. This solution is optimal when assets are easily identified using a single string identifier.

The asset-references element is used to enumerate assets that compose a count in the record. This element is used, instead of identifier-list, when assets are identified using [Asset Identification]. This element has a single attribute that accepts a list of identifiers. Those identifiers must match the identifiers of assets captured in the data-source element. The assets in the data-source element are identifiable using either [Asset Identification], or some other identification scheme. In either case, the identification may be more robust than is permitted with the identifier-list element.

The data-source element is a child of the summary-report element. It has four attributes: id, resource, population-size, and comment. Id is the identifier of the data-source from which a record set is generated. Resource is a URI indicating the actual resource that the data-source element represents. Population-size indicates the number of assets that the resource has knowledge of. Comment allows a comment about the data-source. Data-source has four children: scan-info, extended-info, ai:assets, and asset-list.

The scan-info element is a child of the data-source element. Scan-info is intended to be used when the data-source is a network, vulnerability, compliance, or other type of scan. Scan-info has seven attributes: scan-id, authenticated, execution-location, scan-start, scan-end, population-applies-to, and population-assessed. Scan-id is the ID of the scan. Authenticated represents whether the scan was authenticated or not. Execution-location represents where the scan took place. Scan-start and scanend represent the start and end date and time of the scan. Population-applies-to represents the number of assets that the scan applied to; if a scanner has knowledge of 100 assets, but a particular scan only applies to 50 of them (e.g., a patch scan for a specific OS), the population-applies-to would be set to 50. Population-assessed represents the number of assets that were assessed in the scan. Scan-info has one child, scanner.

The scanner element is a child of scan-info. Scanner has three attributes: product-name, productversion, and scanner-type. The product-name contains the name of the scanner that produced this scan. Product name SHOULD be a CPE name or a SWID, but MAY be any string. Productversion indicates the version of the scanner. Scanner-type indicates the type of scanner. Scanner does not have any children.

The extended-info element is a child of data-source. It is an extension point of the ASR schema, intended to allow data-source information that cannot be captured in other data-source constructs. Extended-info does not have any attributes. Extended-info may have any XML children. The ai:assets element is a child of data-source. It is defined in the [Asset Identification] specification, and is used to list assets. Please see the [Asset Identification] specification for details of this element.

The asset-list element is a child of data-source. It is used to list assets when using ai:assets is not feasible or desired. Asset-list does not have any attributes. Asset-list has one child, asset, which is used to list individual assets.

The asset element is a child of asset-list. Asset has one attribute, id. Id is used to uniquely identify a single asset within the scope

of a data-source. Asset may have any XML children.

6.2 XML Data Model Requirements In order to comply with the ASR data model,

The user MUST produce an XML `asr:summary-report` element consistent with the data model described below.

The XML element produced MUST validate against the XSD for Asset Summary Format 1.0 listed at <http://scap.nist.gov/specifications/asr/#resource-1.0>. In situations where the XSD does not match the documented model in this specification, the XSD SHALL take precedence. The following tables formalize the data model. The data contained in the tables are requirements and MUST be interpreted as follows:

The "Element Name" field indicates the name for the XML element being described. Each element name has a namespace prefix indicating the namespace to which the element belongs. See Table 1-1 for a mapping of namespace prefixes to namespaces.

The "Definition" field indicates the prose description of the element. The definition field MAY contain requirement words as indicated in [RFC 2119].

The "Properties" field is broken into four subfields:

- o The "Name" column indicates the name of a property that MAY or MUST be included in the described element, in accordance with the cardinality indicated in the "Count" field

- o The "Type" column indicates the REQUIRED data type for the value of the property. There are three categories of types: literal, element, and special. A literal type will indicate the type of literal as defined in [XSD]. An element type will reference the name of another element that ultimately defines that property. A special type is listed when the type is neither literal nor element. The special type will indicate the nature of permitted content, such as allowing any XML to be used.

- o The "Count" column indicates the cardinality of the property within the element. The property MUST be included in the element in accordance with the cardinality. If a range is given, and "n" is the upper-bound of the range, then the upper limit is unbounded.

- o The "Definition" column defines the property in the context of the element. The definition MAY contain requirement words as indicated in [RFC 2119].

Element name: asr:summary-report			
Definition	An ASR report may need to be expressed through multiple XML instances. This element is the root of an ASR XML instance. Each ASR XML instance SHALL consist of a single root asr:summary-report element, which encloses a single page of a summary report. An ASR report MAY span one or more pages, indicated by the report-id and page-number properties.		
Properties			
Name	Type	Count	Definition
report-id	NCName	1	The identifier for the report. This SHOULD be unique on a per-report basis. In the case where one report consists of multiple XML documents, this ID MUST match the ID of the related report documents.
page-number	positive integer	1	Which page of the report this XML document represents. If the entire summary report is represented in a single XML document, this attribute MUST be set to "1". If the summary report spans multiple pages, this attribute MUST be set to the positive integer that indicates the page of the current XML document. Each page MUST be numbered sequentially, and

			the sequence MUST start with "1".
last-page	boolean	1	If this XML document represents the last page of a summary report. last-page MUST be set to "true" when this page is the last page, and MUST be set to "false" otherwise. When a summary report is fully represented on one page, last-page MUST be set to "true".
metadata	asr:metadata	0-1	Information about the generation of this asr:summary-report. See Table 6-2.
record-set	asr:record-set	1-n	Information about the record set. See Table 6-3.
data-source	asr:data-source	0-n	A source of data for an asr:summary-report. See Table 6-7.

Note: I will create Tables 6-2 through 6-12 in similar fashion to 6-1 if table 6-1 looks good. Otherwise, it didn't seem worth the time to do all the tables before knowing if the format was appropriate for an RFC.

TODO: Create Table 6-2

TODO: Create Table 6-3

TODO: Create Table 6-4

TODO: Create Table 6-5

TODO: Create Table 6-6

TODO: Create Table 6-7

TODO: Create Table 6-8

TODO: Create Table 6-9

TODO: Create Table 6-10

TODO: Create Table 6-11

TODO: Create Table 6-12

7 Defining a Record Set Type

A record set type is the definition of a record set; it gives context to a record set. A record set type is defined separate from a summary report. It defines the purpose of a record set, the required and optional attributes for each record, whether an asset list should be provided (and if so, which attribute it associates with), and any additional information related to the record set type. This section documents the requirements for defining a record set type.

A record set type may be defined in any form (e.g., verbal, human readable, XML). In whatever form the record set type definition takes, it MUST specify the following information:

A namespace-qualified name, using [XSD Qual] Section 3.2 Qualified Locals, identifying the record set type. This QName is referenced from each record set that implements the record set type. The record set type name is specified in the record-set-type attribute of the record-set element.

The intended use of the record set type.

The namespace-qualified attributes for each record within the record set type. Attributes MUST be namespace qualified as defined using [XSD Qual] Section 3.2 Qualified Locals.

- o For each attribute, it MUST indicate if the attribute MUST, MUST NOT, SHOULD, SHOULD NOT, or MAY be present. For each attribute, it MUST also specify a description of the attribute.

- o If an asset list is permitted or required, a single attribute MUST be designated as the "countfor-asset-list" attribute. The count-for-

asset-list attribute MUST have a type that only allows non-negative integers. A count-for-asset-list attribute SHALL contain an integer that is equal to the number of assets in the asset list for the record. The criteria for an asset to be included in the asset list MUST be specified in the description of the count-for-asset-list attribute. An attribute SHALL NOT be identified as the 'count-for-asset-list' attribute when an asset list is prohibited. At most one attribute on a record set SHALL be specified as the count-for-assetlist attribute.

Whether an asset list is required, permitted, or prohibited. If an asset list is required or permitted, it MUST indicate which attribute it is on the record that the list is associated with.

Whether attributes not explicitly declared in the record set type are permitted.

In the resources section of the ASR website, located at <http://scap.nist.gov/specifications/asr/#resource-1.0>, an XSD provides a common XML format for representing a record set type. Record set types SHOULD be documented in the format defined by the XSD, though they are not required to be. See the XSD for details on documenting the record set type in that format.

Additionally, two Extensible Stylesheet Language Transformations (XSLTs) are provided on the resources section of the ASR website to support record set types documented in the above referenced XSD format. The first XSLT document converts record set type XML documents into a human readable HTML file. The second XSLT accepts an ASR document as input, along with a record set type as a parameter. The XSLT analyzes the ASR document against the provided record set type, and reports any errors that are discovered. Both of the XSLTs, along with the record set type XSD, are informative and are intended to assist with adopting ASR.

8 IANA Considerations

<IANA considerations text>

9 Security Considerations

As a data format, Asset Summary Reporting does not have security concerns that are known at this time. However, as a data format designed to be stored and transmitted between entities within an enterprise, the fact of the matter is that it SHOULD be used within a properly secured environment. Over time, a significant amount of information valuable to attackers can be gleaned from Asset Summary Reporting information. Therefore, it is recommended that use of

Asset Summary Reporting be performed in environments providing communication security mechanisms supplying the properties of confidentiality, data integrity, and non- repudiation.

10 References

10.1 Normative References

- [ARF] NIST Interagency Report (IR) 7694 - Asset Reporting Format 1.1, June 2011. See:
<http://scap.nist.gov/specifications/arf/index.html>
- [Asset Identification] NIST Interagency Report (IR) 7693 - Asset Identification 1.1, May 2011. See:
<http://scap.nist.gov/specifications/ai/index.html>
- [Continuous Monitoring] NIST Special Publication (SP) 800-137 - Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations, January 2012. See:
<http://csrc.nist.gov/publications/PubsSPs.html#800-137>
- [CPE-N] NIST Interagency Report (IR) 7695 - Common Platform Enumeration: Naming Specification 2.3, August 2011. See:
<http://scap.nist.gov/specifications/cpe/naming.html#resource-2.3>
- [RFC 2119] Internet Engineering Task Force (IETF) Request for Comment (RFC) 2119: Key words for use in RFCs to Indicate Requirement Levels, March 1997. See:
<http://www.ietf.org/rfc/rfc2119.txt>
- [XCCDF] NIST Interagency Report (IR) 7275 - Specification for the Extensible Configuration Checklist Description Format 1.2, September 2011. See:
<http://scap.nist.gov/specifications/xccdf/#resource-1.2>
- [XML] W3C Recommendation Extensible Markup Language (XML) 1.0 (Fifth Edition), 26 November 2008. See: <http://www.w3.org/TR/REC-xml/>
- [XSD] W3C Recommendation XML Schema, 28 October 2004. See:
<http://www.w3.org/XML/Schema.html>
- [XSD Qual] W3C Recommendation XML Schema Part 0: Primer Second Edition, 28 October 2004. See:
<http://www.w3.org/TR/xmlschema-0/>

10.2 Informative References

TODO: Decide which references belong in the normative/informative sections

Appendix A Acknowledgements

Special thanks go to Mark Davidson, Adam Halbardier, and David Waltermire, and those others who participated in the definition of the Asset Summary Reporting version 1.0 [IR7848]

Appendix B Use Cases

This appendix documents some common use cases that were considered when developing ASR.

A.1 Continuous Monitoring

Information security continuous monitoring (ISCM) is defined as maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions. Organizational security status is determined using metrics established by the organization to best convey the security posture of an organization's information and information systems, along with organizational resilience given known threat information. Among other requirements, ISCM tools must provide reporting with the ability to tailor output and drill down from high-level, aggregate metrics to systemlevel metrics, and allow for data consolidation into Security Information and Event Management (SIEM) tools and dashboard products.

ASR intends to meet the tool needs above by providing a vendor and technology neutral, and flexible reporting data model. ASR places few constraints on the type and nature of data that can be transported using its model, so many types of continuous monitoring data may be communicated using the model.

A.2 Security Automation

Security automation efforts (e.g. the Security Content Automation Protocol (SCAP)), are another use case for ASR. An example of a security automation need is a system administrator who needs to assess the 15,000 desktops on her network for compliance with the United States Government Configuration Baseline (USGCB), a government baseline for security settings. The system administrator runs the USGCB content against all desktops on her network, and receives one result file per desktop. Each result file contains, for each security

check, a pass/fail indication, required setting, and actual setting. There are over 200 security settings checked per desktop. In order to avoid manually reviewing each result, the system administrator decides to use summary reporting. The system administrator creates three reports, detailed below. A sample record is provided for each report. Note that attributes with the 'example' prefix are defined only in the scope of this use case.

A report that shows the overall compliance percentage of desktops on the network. This summary report is used in a monthly dashboard that is presented to upper management. <asr:record example:compliance_pct="73"/>

Percentage compliance scores for each desktop on the network. This report is used to track perdesktop compliance trends and identify high-risk desktops. <asr:record example:hostname="desktop1" example:compliance_score="100"/>

Percentage compliance scores for the compliance of each security setting across the network. This report is used as a component in taking a risk-based approach to remediation. A bulletin of the top five non-compliant settings is generated monthly and sent to all employees as part of a security awareness campaign. <asr:record asr-attr:xccdf-benchmark="USGCB: Guidance for Securing Microsoft Windows 7 Systems for IT Professional" asr-attr:xccdf-profile="united_states_government_configuration_baseline_version_1.2.0.0" asr-attr:xccdf-rule="minimum_password_length" example:compliance_score="75"/> Since the summary reports are in a standard format, they can be consumed by an application and presented in a meaningful manner. One meaningful presentation of this data is a list of security checks sorted from least compliant to most compliant. The system administrator has used summary reporting to increase the visibility and transparency of her security operations to management and end users, improved the accuracy and completeness of her data, and prioritized her highest value work.

Appendix C Integration with ARF

The Asset Reporting Format (ARF) is a data model for expressing the exchange format of information about assets and the relationships between assets and reports. The data model facilitates the reporting, correlating, and fusing of asset information throughout and between organizations. The intent of ARF is to provide a uniform foundation for the expression of reporting results, fostering more widespread application of sound IT management practices.

ARF has four primary components: assets, reports, report requests, and relationships. Assets, reports, and report requests exist in

isolated buckets. The relationships component allows for explicit relationships between components (assets, report-requests, and reports) and uses a controlled vocabulary to do so. ASR reports may be captured as report objects in ARF. When an ASR is captured as an ARF report payload, a relationship MAY be established between the ASR report and the report request that caused the ASR to be generated. ARF 1.1 defines a relationship type `arf-rel:createdFor` in [ARF] Table 6-1. The `arf-rel:createdFor` relationship, established with the ASR report object as the subject and the report request object as the object, provides a well-defined connection between the request and response. Additional relationships may be defined between ASR reports and other reports as needed. Those relationships should be determined by the report creators and collaborators as needed.

Appendix D Record Set Example

This section shows an example scenario where an ASR report is created. To illustrate the record set concept, consider a scenario where the Chief Information Security Officer (CISO) of Example Corp wants to know the organization's security posture relative to a recently published CVE (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2013>). The following ASR document accurately reports an answer to the CISO's question:

```
<asr:summary-report xmlns:ex="com.example"
xmlns:asr="http://scap.nist.gov/schema/asset-summary-reporting/1.0"
xmlns:asr-attr="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/attr" page-number="1" last-page="true" report-
id="dl1" > <asr:metadata timestamp="2011-11-08T14:27:44.97Z"/>
<asr:record-set id="asr:com.example:rset:1" data-source-
ref="asr:com.example:dsrc:1" record-set-type="ex:cve-report-
small" > <asr:record asr-attr:cve-id="CVE-2011-2013" asr-
attr:inventory-finding="EXISTS" asr-attr:count="50"/> <asr:record
asr-attr:cve-id="CVE-2011-2013" asr-attr:inventory-
finding="NOT_EXISTS" asr-attr:count="170"/> <asr:record asr-
attr:cve-id="CVE-2011-2013" asr-attr:inventory-
finding="NOT_APPLICABLE" asr-attr:count="30"/> </asr:record-set>
<asr:data-source id="asr:com.example:dsrc:1"
resource="VulnDb.abc.com" population-size="250"/> </asr:summary-
report>
```

Notice that the summary report uses attributes defined in the ASR Common Attributes schema. While record set types may use any XML attribute, it is preferable to leverage existing attributes defined in the ASR Common Attributes schema. This is one such example where existing attributes are useful. Since this vulnerability has a CVSS score of 10.0 and a rating of 'Critical' has been given to the patch that fixes this vulnerability, the CISO forwards this information to

the Windows Support Team. The CISO requests that the Windows Support Team apply the appropriate patch quickly. In order to honor the request, the Windows Support Team needs the report broken up by operating system and physical location, with a list of assets. With this information, the Windows Support Team will be able to efficiently deploy their resources for patching. The Windows Support Team requests the report with those additional details (only for systems that need to be patched). The following ASR document accurately reports the needed information. Asset listings have been truncated for simplicity.

```
<asr:summary-report xmlns:ex="com.example" xmlns:ex-
attr="com.example.attr"
xmlns:asr="http://scap.nist.gov/schema/asset-summary-reporting/1.0"
xmlns:asr-attr="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/attr" page-number="1" last-page="true" report-
id="dle2"> <asr:metadata timestamp="2011-11-08T16:33:04.73Z"/>
<asr:record-set id="asr:com.example:rset:2" data-
sourceref="asr:com.example:dsrc:1" record-set-type="ex:cve-report-
informative"> <asr:record asr-attr:cve-id="CVE-2011-2013"
asr-attr:inventory-finding="EXISTS" asr-
attr:count="30" asr-
attr:cpe="cpe:2.3:o:microsoft:windows_7:-:*:*:*:*:*:*"
ex-attr:location="Miami"> <asr:identifier-list identifier-
system="com.example.asset-tag"> <asr:id>111111</asr:id>
<asr:id>222222</asr:id> ... </asr:identifier-list>
</asr:record> <asr:record asr-attr:cve-id="CVE-2011-2013"
asr-attr:inventory-finding="EXISTS" asr-
attr:count="15" asr-
attr:cpe="cpe:2.3:o:microsoft:windows_7:-:*:*:*:*:*:*"
ex-attr:location="Boston"> <asr:identifier-list identifier-
system="com.example.asset-tag"> <asr:id>333333</asr:id>
... </asr:identifier-list> </asr:record> <asr:record asr-
attr:cve-id="CVE-2011-2013" asr-attr:inventory-
finding="EXISTS" asr-attr:count="5"
asr-attr:cpe="cpe:2.3:o:microsoft:windows_2003_server:-
:*:*:*:*:*:*" ex-attr:location="Miami">
<asr:identifier-list identifier-system="com.example.asset-tag">
... </asr:identifier-list> </asr:record> <asr:record asr-attr:cve-
id="CVE-2011-2013" asr-attr:inventory-
finding="EXISTS" asr-attr:count="0"
asr-attr:cpe="cpe:2.3:o:microsoft:windows_2003_server:-
:*:*:*:*:*:*" asr-attr:location="Boston"/>
</asr:record-set> <asr:data-source id="asr:com.example:dsrc:1"
resource="VulnDb.example.com" population-size="250"/> </asr:summary-
report>
```

This information can also be represented as a data table: TODO:

Recreate the table

With the above summary report, the Windows Support Team can deploy the necessary resources to the appropriate locations. With the asset list present in the identifier-list element, the resources deployed to Miami and Boston will be able to accurately and completely remediate the CVE. The report in this section is described in Appendix D.

In this report, Example Corp used attributes defined in the ASR Common Attributes Schema (cve-id, cpe, inventory-finding, and count) as well as an attribute that Example Corp defined, location.

The record sets used in this example have a record-set-type of 'cve-report-small' and 'cve-reportinformative', respectively. While this example uses two record-set-types, it is possible to use a single, more flexible record-set-type. Record-set-types may be flexible or restrictive, depending on the requirements of the entity that defines the record-set-type. Both examples are given in Appendix D

Appendix E Sample Record Set Type Definitions

This section describes an example of creating record set type definitions for the ASR reports described in Appendix C. Those reports leverage two record set types: ex:cve-report-small and ex:cve-reportinformative. The fictitious record set type cve-report-small is illustrated below. To demonstrate the flexibility of record set type definitions, a more permissive record set type is included in this section below the ex:cve-report-small example.

Record Set Type Name: {com.example}cve-report-small Description: To report on the number of computers affected by a CVE. Attributes

asr-attr:cve-id - MUST include. This is the CVE ID being reported on. Type: XML schema "string".

asr-attr:inventory-finding - MUST include. This is a status of the CVE for each asset in the count. Value must be one of "EXISTS", "NOT_EXISTS", "NOT_APPLICABLE", "NOT_REPORTED", "ERROR", or "UNKNOWN". Type: XML schema "string".

asr-attr:count - MUST include. Asset list is associated with this attribute. This count is the number of assets with the CVE related to the asset via the inventory-finding. Type: XML schema nonNegativeInteger.

Permit attributes not explicitly described here: no

Require asset list: not permitted

Require identifier list: not permitted

The record set type documented above may be more formally represented using the ASR record set type XML definition format:

```
<record-set-types xmlns="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/record-set-type" xmlns:ex="com.example"
xmlns:asr="http://scap.nist.gov/schema/asset-summary-reporting/1.0"
xmlns:asr-attr="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/attr"> <record-set-type record-set-type-
qname="ex:cve-report-small" permitadditional-attributes="false">
<description> To report on the number of computers affected by a CVE.
    </description> <attributes>      <attribute attribute-qname="asr-
attr:cve-id" use="MUST"      description="This is the CVE ID being
reported on"/>      <attribute attribute-qname="asr-attr:inventory-
finding" use="MUST" description="This is a status of the CVE for each
asset in the count. Value must be one of 'EXISTS', 'NOT_EXISTS',
'NOT_APPLICABLE', 'NOT_REPORTED', 'ERROR', or 'UNKNOWN'." />
<attribute attribute-qname="asr-attr:count" use="MUST"
description="Asset list is associated with this attribute. This count
is the number of assets with the CVE related to the asset via the
inventoryfinding."/>      </attributes> <asset-listing use-identifier-
list="MUST NOT" use-asset-references="MUST NOT"/> </record-set-type>
</record-set-types>
```

Given the definition above, the first report in Appendix C is able to be produced. The description above documents the following:

The record set type name "ex:cve-report-small".

The required attributes for the report. For each attribute, it specifies that the attribute must be

included. The attribute type is defined outside the record set type definition.

There are not any attributes permitted in addition to those specified here.

No form of asset listing is allowed.

The XML representation of the record set type definition (as described in Section 7) for ex:cve-reportinformative is illustrated below. <record-set-types xmlns="http://scap.nist.gov/schema/asset-summaryreporting/1.0/record-set-type" xmlns:ex="com.example" xmlns:asr="http://scap.nist.gov/schema/asset-summary-reporting/1.0"

```

xmlns:asr-attr="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/attr"  xmlns:ex-attr="com.example.attr">
<record-set-type record-set-type-qname="ex:cve-report-informative"
permitadditional-attributes="false">  <description> To report on
the number of computers affected by a CVE.  </description>
<attributes>  <attribute attribute-qname="asr-attr:cve-id"
use="MUST"  description="This is the CVE ID being reported
on"/>  <attribute attribute-qname="asr-attr:inventory-finding"
use="MUST" description="This is a status of the CVE for each asset in
the count. Value must be one of 'EXISTS', 'NOT_EXISTS',
'NOT_APPLICABLE', 'NOT_REPORTED', 'ERROR', or 'UNKNOWN'." />
<attribute attribute-qname="asr-attr:cpe" use="MUST" description="The
Common Platform Enumeration for the operating system of the
applicable assests." />  <attribute attribute-qname="ex:location"
use="MUST" description="The physical location of the applicable
assests." />  <attribute attribute-qname="asr-attr:count"
use="MUST" description="Asset list is associated with this attribute.
This count is the number of assets with the CVE related to the asset
via the inventoryfinding." count-for-asset-list="true"/>
</attributes>  <asset-listing use-identifier-list="MUST" use-asset-
references="MUST NOT"/> </record-set-type> </record-set-types>

```

The description above documents the following:

The record set type name "ex:cve-report-informative".

The required attributes for the report. For each attribute, it specifies that the attribute must be included. The attribute type is defined outside the record set type definition.

There are not any attributes permitted in addition to those specified here.

Assets must be listed using the identifier-list method. It is possible for a record set type definition to be flexible enough that both reports in Appendix C could be produced in compliance with it. That record set type definition is illustrated below:

```

<record-set-types xmlns="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/record-set-type"  xmlns:ex="com.example"
xmlns:asr="http://scap.nist.gov/schema/asset-summary-reporting/1.0"
xmlns:asr-attr="http://scap.nist.gov/schema/asset-
summaryreporting/1.0/attr">  <record-set-type record-set-type-
qname="ex:cve-report-informative" permitadditional-attributes="true">
  <description> To report on the number of computers affected by a
CVE.  </description> <attributes>  <attribute attribute-
qname="asr-attr:cve-id" use="MUST"  description="This is the
CVE ID being reported on"/>  <attribute attribute-qname="asr-

```

```
attr:inventory-finding" use="MUST" description="This is a status of
the CVE for each asset in the count. Value must be one of 'EXISTS',
'NOT_EXISTS', 'NOT_APPLICABLE', 'NOT_REPORTED', 'ERROR', or
'UNKNOWN'." />      <attribute attribute-qname="asr-attr:count"
use="MUST" description="Asset list is associated with this attribute.
This count is the number of assets with the CVE related to the asset
via the inventoryfinding." count-for-asset-list="true"/>
</attributes> <asset-listing use-identifier-list="OPTIONAL" use-
asset-references="MUST NOT"/> </record-set-type> </record-set-types>
```

The description above documents the following:

The record set type name "ex:cve-report-informative".

That attributes beyond those listed in the record set type are permitted in record sets. Therefore, in the second example in Appendix C, "asr-attr:cpe" and "ex-attr:location" are permitted. This is known because of the value of "permit-additional-attributes".

The required attributes for the report. For each cve-id, inventory-finding, and count, it specifies that the attribute must be included.

The attribute associated with the asset list. In this case, "asr-attr:count" is the count associated with the asset listing.

The use of asset listings. In this case, the report may optionally use the identifier list (which the second report in Appendix C does), but it may not use the asset references element.

All of the information needed to properly format the aforementioned record sets is included in the record set type definitions given above.

This section contains a list of XML attributes defined in the ASR Common Attributes XSD located at <http://scap.nist.gov/specifications/asr/#resource-1.0>. These attributes are defined to provide a core of usable attributes with a common meaning across multiple areas. Each attribute is accompanied with a short description that describes what it means. Usage of this XSD and its associated attributes is RECOMMENDED, but not required.

Some attributes may require additional context in order to make sense. For example, the statistical 'count' attribute has little meaning by itself. The context can be provided in the record-set-type definition through the report's description of the attribute. For example, if a record-set-type definition had the following text in

the attribute definition, the count would make sense: "Contains the count of employees that have completed the annual security awareness training."

All attributes in this section are defined in the following namespace: `http://scap.nist.gov/schema/assetsummary-reporting/1.0/attr`. Types specified with the prefix "xs:" are XML schema datatypes as defined in [XSD] Part 2. Unless otherwise specified, values for all attributes are restricted to `xs:string`. All pattern restrictions are XML schema regular expressions as defined in [XSD] Part 2, Section G

F.1 SCAP Attributes

This section contains SCAP attributes that are intended for use in reporting scenarios that involve SCAP data.

TODO: Create this table

F.3 Statistical Attributes

This section contains common statistical attributes with well-defined mathematical meanings.

TODO: Create this table

F.4 Other Attributes

This is a list of attributes that do not fit into another category. Some of these attributes exist to provide a common name in attempt to provide interoperability and may be further restricted as reporting scenarios dictate.

TODO: Create this table

E.2 Finding Attributes

This section contains attributes related to findings. Each finding attribute has values that allow the state of the finding to be indicated as well as attributes that allow for the indication of various non-finding conditions (error, not applicable, etc.)

TODO: Create this table

Authors' Addresses

Mark Davidson

EMail: mdavidson@mitre.org

David Oliva

EMail: david.oliva@verizon.net

Network Working Group
Internet Draft
Intended status: Informational
Expires: April 2013

S. Hanna
Juniper Networks
October 15, 2012

Standard Assessment Protocols and Formats for SACM
draft-hanna-sacm-assessment-protocols-00.txt

Abstract

The draft charter for the SACM BOF at IETF 85 calls for the development of "continuous assessment interfaces". This draft points out several existing documents that provide a good start in this area.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction.....	2
2. Languages and Enumerations.....	2
3. Protocols.....	3
4. Merging The Two.....	4
5. Next Steps.....	4
6. Security Considerations.....	5
7. IANA Considerations.....	5
8. References.....	5
8.1. Informative References.....	5
9. Acknowledgments.....	6

1. Introduction

The draft charter for the SACM BOF at IETF 85 [1] calls for the development of "continuous assessment interfaces". This text from the draft charter provides more detail about what's desired:

2. Define, either by normative reference, adoption, or creation, a set of standards that can be used to continuously assess and report on the state of systems, composed of many different types of devices and networks, operated by varying personnel, to ensure security process effectiveness in a pre-defined or ad-hoc manner. This area of focus provides for integration protocols supporting plug and play continuous assessment and security automation networking within an enterprise.

Actually, there are several specifications from IETF and other organizations that provide a very good start on addressing this problem. More work is certainly needed but the SACM BOF should be aware of these documents.

2. Languages and Enumerations

The SCAP specification [2] lists a large number of languages and enumerations that are useful for remote security assessment: XCCDF [3], OVAL [4], OCIL [5], Asset Identification [6], CCE [7], CPE [8], and CVE [9]. Since there is a great deal of implementation

experience with these specifications, they should certainly be considered by the SACM BOF or any successor Working Group.

3. Protocols

The IETF NEA Working Group has defined an architecture and a layered set of protocols for remote assessment of endpoint security posture: the NEA Architecture [10], PA-TNC [11], PB-TNC [12], PT-TLS [13], and PT-EAP [14]. These protocols are designed to be used either at the time that an endpoint connects to a network or continuously after the endpoint is connected to the network.

The NEA protocols are based on the Trusted Network Connect (TNC) protocols, which were created by the Trusted Computing Group (TCG) and donated to the IETF. The TCG contributed the TNC specifications to the IETF in full compliance with BCP 78 [15] and BCP 79 [16], transferring change control and copyright to the IETF (among other things). The IETF took full advantage of this change control, adopting the TNC standards through an open and competitive process but adapting them to the IETF's needs and processes. For example, the IETF renamed all the TNC protocols: IF-M became PA-TNC, IF-TNCCS became PB-TNC, IF-T Binding for TLS became PT-TLS, and IF-T Binding for Tunnelled EAP Methods became PT-EAP.

Because the NEA protocols are based on the TNC protocols, they benefit from the experiences of and feedback from millions of users, thousands of customers, and dozens of vendors and open source implementers who have used the TNC protocols. For example, users strongly prefer quick and efficient checks when waiting to get on the network. Therefore, all the NEA protocols use a binary encoding and minimize round trips. Still, vendors need extensibility so the NEA specs permit vendor-specific extensions while requiring that vendors work without them.

Two of the NEA protocols (PA-TNC and PB-TNC) were published as Proposed Standards in 2010. At the same time, the TCG issued updated TNC protocol specs (IF-M 1.0 [17] and IF-TNCCS 2.0 [18]) that correspond exactly to the NEA specs, thus ensuring that the two architectures remain in alignment. The other two NEA specs (PT-TLS and PT-EAP) are expected to be published as Proposed Standards within the next few months. TCG may reasonably be expected to again issue updated versions of the corresponding TNC specs to maintain alignment. Customer and vendor adoption is expected to be rapid for these specs since the old versions were widely implemented and the new specs are a simple upgrade from the old. Even vendors who have long used proprietary protocols have indicated their plans to support the new open standard protocols.

4. Merging The Two

While the NEA protocols define the format for some simple posture checks (anti-virus or host firewall status, OS patch level), they do not define standards that approach the level of detail that sophisticated enterprise customers need and can achieve with SCAP.

At the same time, SCAP does not define any standards for gathering SCAP content from an endpoint. This is left to the vendor, resulting in a situation where each vendor must place a software agent on the endpoint in order to assess that endpoint (or settle for an external scan, which has lower fidelity).

What's needed to fully satisfy the SACM BOF's charter item on continuous assessment interfaces is a standard for conveying the SCAP languages and enumerations in the NEA protocols.

Fortunately, the TCG has recently published exactly this document. The TCG's SCAP Messages for IF-M specification [19] was published for Public Review on TCG's web site on October 3, 2012. This document describes how SCAP content should be carried over the NEA (TNC) protocols. It includes support for provisioning SCAP content to endpoints, for rapidly and efficiently gathering assessment results when a device connects to the network, for gathering exhaustive information in the background after the device is connected to the network, and for continuously monitoring changes to configuration.

While the TCG has not made any official statements about its intent with respect to donating this specification to IETF, I believe that the TCG would be glad to do so if the IETF charters a Working Group to work on continuous assessment interfaces. I should know about this. I'm co-chair of the TCG's Trusted Network Connect Work Group.

5. Next Steps

The SACM BOF participants should review the new SCAP Messages for IF-M specification to see if it meets their needs. If they find deficiencies, they should notify the TCG by sending email to SCAP-Messages-Comments@trustedcomputinggroup.org.

Within IETF, we should review and discuss these documents to see if they are relevant to the SACM effort. Do they meet the need for continuous assessment interfaces that was described in the proposed SACM charter? If not, what changes are needed? Could those changes be made using these specs as a starting point, assuming that the TCG donated the specs to the IETF with all rights and full change

control? And should this work happen in a new Working Group or should it happen in the NEA Working Group, which already has five years of experience with this topic.

The IETF discussions should happen on the sacm@ietf.org list. I would also be glad to lead a discussion of this topic at the SACM BOF at IETF 85.

6. Security Considerations

This document describes several existing standards relating to endpoint assessment and configuration management. Each of these specifications includes its own Security Considerations section so the reader is referred to those documents for more details.

7. IANA Considerations

This document has no actions for IANA.

8. References

8.1. Informative References

- [1] SACM BOF Draft Charter, <http://www.ietf.org/mail-archive/web/sacm/current/msg00628.html>
- [2] U.S. NIST, "The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2", NIST Special Publication 800-126 Revision 2, September 2011.
- [3] U.S. NIST, "Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2", NIST Interagency Report 7275 Revision 4, September 2011.
- [4] The MITRE Corporation, "The OVAL(R) Language Specification Version 5.10.1", January 2012.
- [5] U.S. NIST, "Specification for the Open Checklist Interactive Language (OCIL) Version 2.0", NIST Interagency Report 7692, April 2011.
- [6] U.S. NIST, "Specification for Asset Identification 1.1", NIST Interagency Report 7693, June 2011.
- [7] The MITRE Corporation, <http://cce.mitre.org>
- [8] U.S. NIST, <http://scap.nist.gov/specifications/cpe>

- [9] The MITRE Corporation, <http://cve.mitre.org>
- [10] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, June 2008.
- [11] Sangster, P., and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5792, March 2010.
- [12] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5793, March 2010.
- [13] Sangster, P., N. Cam-Winget, and J. Salowey, "PT-TLS: A TCP-based Posture Transport (PT) Protocol", draft-ietf-nea-pt-tls-07.txt (work in progress), August 2012.
- [14] Cam-Winget, N. and P. Sangster, "PT-EAP: Posture Transport (PT) Protocol For EAP Tunnel Methods", draft-ietf-nea-pt-eap-03.txt (work in progress), July 2012.
- [15] Bradner, S. and J. Contreras, "Rights Contributors Provide to the IETF Trust", RFC 5378, November 2008.
- [16] Bradner, S., "Intellectual Property Rights in IETF Technology", RFC 3979, March 2005.
- [17] Trusted Computing Group, "IF-M: TLV Binding", Version 1.0, Revision 37, March 2010.
- [18] Trusted Computing Group, "IF-TNCCS: TLV Binding", Version 2.0, Revision 16, January 2010.
- [19] Trusted Computing Group, "SCAP Messages for IF-M", Version 1.0, Revision 16, October 2012.

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Author's Address

Steve Hanna
Juniper Networks, Inc.
79 Parsons Street
Brighton, MA 02135
USA
Email: shanna@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 27, 2013

A. Montville, Ed.
Tripwire, Inc.
September 23, 2012

Asset Identification
draft-montville-sacm-asset-identification-00

Abstract

Asset identification plays an important role in an organization's ability to quickly correlate different sets of information about assets. This document provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information about the assets. This document describes the purpose of asset identification, a data model for identifying assets, methods for identifying assets, and guidance on how to use asset identification. It also identifies a number of known use cases for asset identification.

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 27, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

One of the primary requirements for performing asset management is the ability to identify assets based on some set of data known about them. Asset identification, the use of attributes and methods to uniquely identify an asset, allows for correlation of data across multiple sources, reporting of asset information across different organizations and databases, targeted actions against specific assets, and usage of asset data in other business processes.

Unfortunately, neither a unified method nor a published specification for performing asset identification exists at this time. Existing security automation specifications either do not consider asset identification or represent identification information differently than other specifications with which they interoperate. This means that correlation of data relies on a transformation process between each specification, which is expensive and unreliable. Creation of such a unified method and specification for performing asset identification would allow for greater interoperability, increased capabilities, and easier implementation of asset management processes.

This Asset Identification specification describes a framework for how asset management processes and other specifications may identify assets using some set of information known or generated about the asset. It describes the data model and representation of asset identification information and it provides requirements for consuming and producing identification information. Requirements for usage of asset information and requirements for how the information that identifies assets is collected or generated are out of scope for this specification.

For the purposes of this specification, an asset is considered to be anything that has value to an organization. For example, computing devices are one form of asset that many organizations track. This specification, however, does not limit asset identification to identifying computing devices; any type of asset may be identified. The specification itself provides constructs for identifying many types of assets, and users may extend the model to include other asset types if they wish to identify asset types that are not addressed in the specification.

It is expected that other standards, data formats, tools, processes, and organizations will reference this specification to describe how to represent asset identification information. This will ensure compatibility of asset identifications among these components and allow for improved asset management processes.

While this specification was developed to support the immediate needs of the security automation community, it is expected that it will be valuable in general asset management processes both inside and outside of the security automation space.

1.1. Purpose and Scope

The purpose of this document is to define the Asset Identification specification, a standardized model for representing and identifying assets.

The scope of this document is to give an introduction to Asset Identification, give guidelines on using Asset Identification, describe the Asset Identification data model, and document conformance requirements to comply with Asset Identification. Other versions of Asset Identification and the associated component specifications, including emerging specifications and future versions, are not addressed here.

Future versions of Asset Identification will be defined in distinct revisions of this document, each clearly labeled with a document revision number and the appropriate Asset Identification version number.

1.2. Audience

This specification is intended for authors of specifications that must support asset identifications, implementers of those specifications, system integrators composing architectures from tools that implement those specifications, and end users who wish to understand how these tools work.

1.3. Document Structure

The remainder of this document is organized into the following major sections:

- o Section 2 defines the terms used within this specification and provides a list of common abbreviations.
- o Section 3 describes how this specification fits with related standards and specifications.
- o Section 4 defines the conformance requirements for asset identification.
- o Section 5 gives an overview of asset identification.

- o Section 6 describes the asset identification data model constructs.
- o Section 7 presents the asset identification schema.
- o Appendix A recognizes the work of the original Asset Identification specification [IR7693].
- o Appendix B describes possible use cases for asset identification.
- o Appendix C explains how the specification can be extended.

1.4. Document Conventions

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 [RFC2119].

Both inline and indented forms use qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the specific XML schema that defines (and consequently may be used to validate) the syntax of the element instance. A qualified name declares this schema to element association using the format prefix:element-name. The association of prefix to namespace is defined in the metadata of an XML document and generally will vary from document to document. In this specification, the conventional mappings listed in Table 1-1 are used.

Prefix	Namespace URI	Schema
ai	urn:ietf:params:xml:ns:asset-identification-1.0	Asset Identification 1.0
core	http://scap.nist.gov/schema/reporting-core/1.1	SCAP Reporting Core 1.1
cpe-name	http://cpe.mitre.org/naming/2.0	CPE 2.3 Naming Specification
xal	urn:oasis:names:tc:ciq:xsd:schema:xAL:2.0	OASIS extensible Address Language

xnl	urn:oasis:names:tc:ciq:xsdschema:xNL:2.0	OASIS	
		extensible	
		Name Language	
+-----+	+-----+	+-----+	+-----+

Conventional XML Mappings

2. Terms and Abbreviations

Asset: Anything that has value to an organization, including, but not limited to, another organization, person, computing device, information technology (IT) system, IT network, IT circuit, software (both an installed instance and a physical instance), virtual computing platform (common in cloud and virtualized computing), and related hardware (e.g., locks, cabinets, keyboards).

Asset Identification: The use of attributes and methods to uniquely identify an asset.

Asset Identification Element: A complete, bound expression of an asset identification using the constructs defined in this specification.

Circuit: A dedicated single connection between two endpoints on a network.

Computing Device: A machine (real or virtual) for performing calculations automatically (including, but not limited to, computer, servers, routers, switches, etc.)

Data: Any piece of information suitable for use in a computer.

Database: A repository of information or data, which may or may not be a traditional relational database system.

Extension Identifier: Any piece of identifying information provided in an asset identification element that is not explicitly defined in the Asset Identification schema.

Identifying Information: The set of an asset's attributes that may be useful for identifying that asset, including discoverable information about the asset and identifiers assigned to the asset.

Matching: The process of determining whether two or more asset identification expressions refer to the same asset.

Network: An information system(s) implemented with a collection of interconnected components. Such components may include routers, hubs, cabling, telecommunications controllers, key distribution centers, and technical control devices.

Organization: An entity of any size, complexity, or positioning within an organizational structure (e.g., a federal agency, or, as appropriate, any of its operational elements).

Person: Any person considered as an asset by the management domain.

Relationship Identifier: Identifying information where the value is a relationship to another asset.

Service: A set of related IT components provided in support of one or more business processes.

Software: Computer programs and associated data that may be dynamically written or modified during execution.

System: A discrete set of information resources organized for the collection, processing, maintenance, use, sharing, dissemination, or disposition of information.

Synthetic Identifier: An identifier that is assigned to an asset in the context of some management domain.

Website: A set of related web pages that are prepared and maintained as a collection in support of a single purpose.

2.1. Acronyms

BIOS: Basic Input/Output System

CIDR: Classless Inter-Domain Routing

CPE: Common Platform Enumeration

FQDN: Fully-Qualified Domain Name

GUID: Globally Unique Identifier

HTTP: Hypertext Transfer Protocol

IETF: Internet Engineering Task Force

IP: Internet Protocol

IT: Information Technology

ITL: Information Technology Laboratory

MAC: Media Access Control

NIST: National Institute of Standards and Technology

OASIS: Organization for the Advancement of Structured Information
Standards

RFC: Request for Comment

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

W3C: World Wide Web Consortium

WFN: Well-Formed Name

xAL: Extensible Address Language

XML: Extensible Markup Language

xNL: Extensible Naming Language

XSD: XML Schema

3. Relationship to Existing Standards and Specifications

This specification defines the constructs and methods for representing asset identification information and thus can be leveraged by any other specification where identifying assets is required or beneficial.

This specification uses several industry-standard mechanisms for representing identification information and providing conformance requirements.

Common Platform Enumeration (CPE)TM is a structured naming scheme for information technology systems, platforms, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format. CPE version 2.3 Well-Formed Names (WFN) are used as software- identifying information by this specification [CPE23].

The extensible Address Language (xAL) by the Organization for the Advancement of Structured Information Standards (OASIS) is an XML standard format for representing international address information [XAL]. Asset Identification leverages xAL to represent address information for assets.

The extensible Name Language (xNL) by OASIS is an XML standard format for representing the names of people and organizations. [XNL] Asset Identification leverages xNL to represent the names of people and organizations.

4. Conformance

A product may want to claim conformance with this specification so that users and organizations can use the product with the assurance that the product can identify assets in a consistent and standard manner. The ability for a product to identify assets in a standard manner increases the likelihood of interoperability between conforming products. This section defines the criteria for products to claim conformance with this specification.

4.1. Product Conformance

Products are divided into two roles based on their use of asset identification information: consumers and producers.

- o Consuming products ("consumers") must be able to receive and understand information in compliance with this specification.
- o Producing products ("producers") must create asset identification information in a format compliant with this specification.

A product may be both a consumer and producer. The following subsections document the conformance requirements for the two types of products.

4.1.1. Consumers

Any consuming product claiming conformance to this specification MUST adhere to the following requirements.

- o The consumer SHALL be capable of processing the identification information represented in constructs consistent with the Asset Identification data model without error. See Section 6.
- o The consumer MAY attempt to consume constructs that are invalid per the Asset Identification data model. See Section 6.
- o The consumer MAY consume extension identifiers and use them as an input into a matching process. See Section 5.3.4.
- o The consumer MUST NOT abnormally end, crash, or otherwise be unable to fully process asset identification elements that include extension identifiers. It MAY ignore any information in extension identifiers.

4.1.2. Producers

Any producing product claiming conformance to this specification MUST adhere to the following requirements.

- o The producer SHALL accurately produce the asset identification element in XML consistent with the data model. Section 6.
- o When representing identification information, the producer SHOULD provide as much information as is sufficient to allow for a match. See Section 5.3.
- o When representing identification information, the producer MAY provide as much or as little identifying information as allowed in the data model per other recommendations or tool capabilities. See Section 5.3.
- o The producer MAY provide extension identifiers for any asset identification element. See Section 5.3.4.

5. Asset Identification Overview

This section gives an overview of the Asset Identification model and its key concepts.

5.1. Scope

In order to support the variety of use cases discussed in Appendix A, the scope of this specification is limited to a description of how asset management tools can represent asset identification information when communicating it to other tools. It is out of scope of this specification to recommend which identifiers to use or to require that identification information be collected in a certain way or from a certain place. Higher-level specifications, tools, and organizations that implement Asset Identification, however, are encouraged to make these recommendations or specify these requirements in order to support the particular needs of their use cases.

Additionally, the Asset Identification specification is not a mechanism for expressing information about an asset that is not related to asset identification. Only elements that are used for identification are included in the core specification. Asset Identification elements **MUST NOT** be used to represent information about an asset unless it is being used to identify that asset.

5.2. Core Specification and Extension Points

The core Asset Identification specification defines eleven asset types and definitions of how those asset types may be identified using a set of literal attributes and relationships to other assets. The core specification is intended to provide definitions for commonly used asset types and identification attributes; it is not intended to be an exhaustive list of all possible asset types and attributes that may be used for identification. Anything explicitly defined in the asset identification schema and the asset identification controlled vocabulary for relationship identifiers is considered part of the core specification.

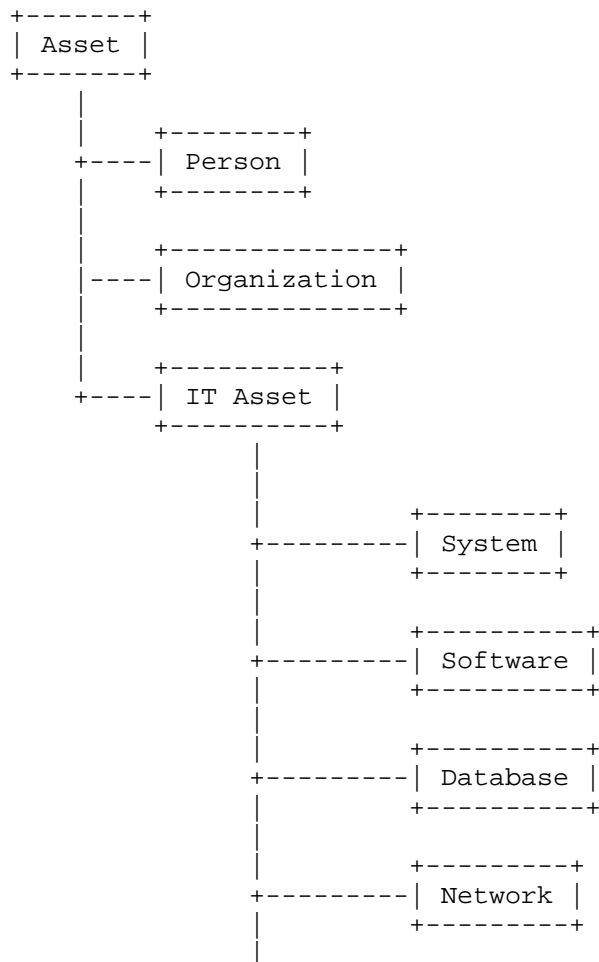
There are several extension points in the Asset Identification data model to allow for identification of asset types beyond what is included in the core specification and to allow attributes or relationships outside of the core specification to be used to identify asset types that are included. These extension points are:

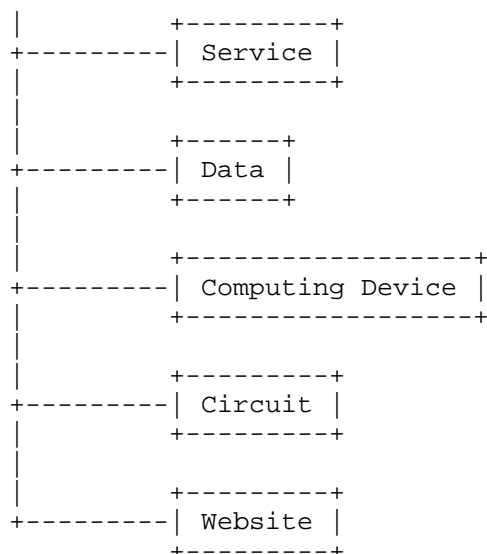
- o Additional asset types may be created by inheriting from any concrete or abstract "asset" data element in the core XML schema.

- o The core asset types may be enhanced by adding elements to the appropriate asset type as literal values in the "extended-information" element.
- o Additional relationships can be defined by creating a separate vocabulary for relationship identifiers.

5.3. Data Model Overview

The Asset Identification data model consists of a set of asset types and a set of information that can be provided about each asset type. The asset types currently supported in this specification are:





For the purposes of this specification the above asset types MUST be understood as defined in Section 2.

The specification MAY be extended by Asset Identification producers to allow for other asset types as needed; however, it is OPTIONAL for Asset Identification consumers to support asset types not present in the core specification.

For each asset type above, the specification has a core set of fields that may be provided in order to identify an asset of that type. For example, an asset of type "person" may be identified by an email address, full name, telephone number, or birth date. Any number of these fields may be populated in order to create an asset identification element. Specifications, management environments, organizations, and tool vendors implementing Asset Identification are encouraged to recommend, restrict, or require that certain fields be populated or not populated; however, the specification itself does not do so.

There are several different types of information that may be used to identify assets: literal identifiers, relationship identifiers, synthetic identifiers, and extension identifiers. These four identifier types are differentiated only because they are represented differently in the data model. No identifier type is intrinsically more or less valuable for performing asset identification than any other identifier type.

5.3.1. Literal Identifiers

Literal identifiers are the pre-defined fields containing literal values that may identify an asset. For example, Media Access Control (MAC) address is an example of a literal identifier for a computing device. Literal identifiers defined in Asset Identification **MUST** be properly processed without error by Asset Identification consumers.

5.3.2. Synthetic Identifiers

Synthetic identifiers are meant to be used when a database or process assigns an identifier. For example, an employee is often assigned an employee identifier which may be used to track him or her across the organization. These identifiers should be represented using the synthetic identifier construct: the namespace denotes the management domain for which the identifier is valid and the identifier contains the identifier itself. Each asset type allows for a list of zero to many synthetic identifiers. Synthetic identifiers **MUST** be properly processed without error by Asset Identification consumers.

5.3.3. Relationship Identifiers

Relationship identifiers are meant to be used when an asset may be identified based on a relationship to another asset. For example, a system may be identified based on the fact that it is named "System 1" and it is connected to network "INTERNAL". Relationship types are represented as a controlled vocabulary. Any relationships defined in the Asset Identification controlled vocabulary are core and **MUST** be processed without error by Asset Identification consumers. Relationships that are defined in other controlled vocabularies are considered extension identifiers and **MAY** be supported by Asset Identification consumers.

5.3.4. Extension Identifiers

Although this specification intends to support the most common types of information that are used to identify assets, certain users, organizations, or use cases may find that the core model does not support some fields that they need. Asset Identification supports a producer's ability to provide these identifiers in any asset identification element through extension identifiers; however, it is **OPTIONAL** for consumers to process or understand these identifiers unless some other specification requires it. Extension identifiers may include additional literal values as well as relationship identifiers that are outside of the Asset Identification controlled vocabulary. Extension identifiers **MUST** be processed without error by consumers; however, consumers are encouraged to ignore identifying information that they do not understand and is not defined in the

core schema, which ensures accurate correlation.

5.4. Providing Asset Identifications

In the absence of other guidance or requirements, Asset Identification providers SHOULD provide as much information as they have available in the core (non-extension) asset identification element. Bandwidth constraints, other specifications, and tool intelligence MAY help define how much or which information SHOULD be provided beyond this recommendation.

5.5. Consuming Asset Identifications

Asset Identification consumers MUST be able to process literal identifiers, synthetic identifiers, relationship identifiers, and extension identifiers without error. In this context, "process" simply means ingest without error and optionally use as an input in performing a matching. Asset Identification consumers SHOULD process literal identifiers, synthetic identifiers, and relationship identifiers and support incorporating them into a matching process. Asset Identification consumers SHOULD NOT incorporate unknown extended identifiers into a matching process as they may be misleading or misunderstood.

Extended identifiers that are defined in another specification or policy that the consumer implements MAY be supported as appropriate and as defined by that specification or policy.

5.6. Matching

Matching is the process of determining whether or not two or more asset identification elements are referring to the same asset. Matching is performed across an entire asset identification element, not across each individual property of an identifier.

Although matching identifiers is an important part of the asset identification process, due to a wide variety of current tool practices, organizational architectures, and the need to allow for innovation, this specification does not provide any normative requirements in regards to matching. Tools are free to perform matching based on their own logic, and specifications implementing asset identification are encouraged to provide their own recommendations or requirements around matching.

5.7. Sample Correlation Workflow

The diagram in Figure 1 shows a sample correlation workflow, including matching discoverable information and several synthetic

identifiers.

In this sample architecture, which is merely one example, several tools report on information about an asset. The information is correlated by an asset database, potentially processed or aggregated, and then reported to a higher-level database.

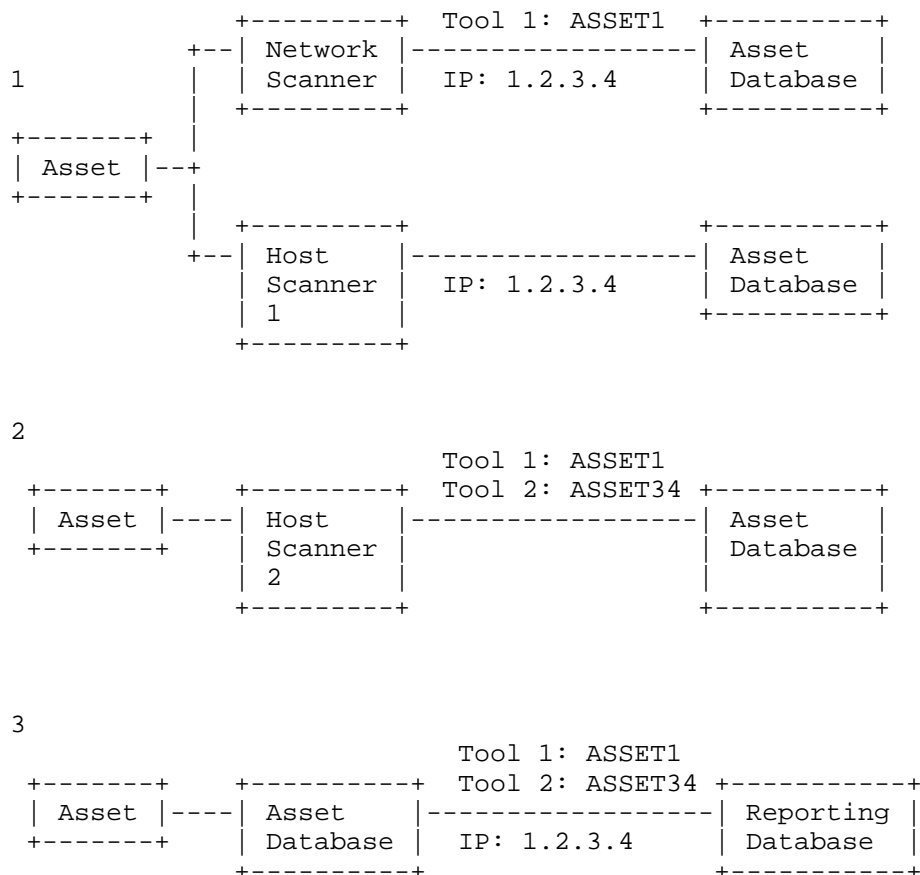


Figure 1: Sample Correlation Workflow

In step 1, a host-based scanner is reporting on asset information (e.g. vulnerability assessment results) using a synthetic identifier in its own namespace and an IP address as identifying information. Additionally, a network scanner is reporting on network events by IP address. This allows the asset database to correlate information coming from the network with information on the host, potentially matching vulnerabilities discovered by the host-based scanner with attacks against that vulnerability discovered by the network scanner.

In step 2, another host-based scanner (e.g., an asset inventory tool) reports data using both a synthetic identifier in its own namespace and a synthetic identifier in the first host-based tool's namespace. This other identifier may have been collected on the system or may have been discovered some other way; how that collection happens is out of scope of this specification. By passing both identifiers, however, the scanner provides enough data to the asset database to correlate the additional inventory data with the vulnerability and event data. Additionally, any other data reported using either the IP address or the two synthetic identifiers will be able to be correlated as well. Note that this specification does not require or recommend that tools process identifiers in certain ways: for example, an IP address may become stale after a period of time, but it is out of scope for this specification to recommend how to deal with that.

In step 3, the asset database provides a report to a higher-level database. Depending on the reporting architecture, organizational hierarchy, and reporting requirements, asset databases may want to report on data with different sets of synthetic identifiers for each asset. This specification does not restrict or recommend architectures or workflows.

In the reporting architecture shown above, all known asset identifiers for each asset are being used to report information to the higher level. In this architecture, data provided by other tools to the higher-level database may be correlated with the reported data. Other options would be to only report some information to the higher-level database or even to generate a new synthetic identifier to perform reporting, depending on the reporting requirements and network architectures.

6. Data Model

This section documents the data model for Asset Identification. The XML schema that implements this data model is provided separate from this specification.

The Asset Identification model is a fairly flat model that defines the constructs to hold identifying information about an asset. A limited number of asset types are defined for which model constructs exist. In order to use the Asset Identification model:

- o The user SHOULD produce an XML ai:assets or ai:asset-related element consistent with the data model described in Section 6.3.12.1.
- o The XML element produced MUST validate against the XML schema (XSD) for Asset Identification in Section 7. In situations where the XML schema does not match the documented model in this specification, the XSD takes precedence.

The following subsections formalize the logical data model. The data contained in the subsections include prose and tables, which MUST be interpreted as follows:

- o The subsection title indicates the name for the entity being described.
- o The "INHERITS" field indicates that the element takes on all of the properties of the inherited element in addition to the properties defined for the element.
- o The "DEFINITION" field indicates the prose description of the text. The field MAY contain requirement words as indicated in [RFC2119].
- o The "Properties" field is broken into a four column table:
 - * The "Name" column indicates the name of a property that MAY or MUST be included in the described element in accordance with the cardinality indicated in the "Count" field.
 - * The "Type" column indicates the type of data that MUST be the value for the property. There are two categories of types: literal and element. A literal type will indicate the type of literal. The element type will reference the name of another element that defines the content for that property.

- * The "Count" column indicates the cardinality of the property within the element. The property MUST be included in the element in accordance with the cardinality. If a range is given, and "n" is the upper-bound of the range, then the upper limit is unbounded.
- * The "Definition" column defines the property in the context of the element. The field MAY contain requirement words as indicated in [RFC2119].

Each literal data element MAY have a "source" attribute associated with it. The source attribute is intended to capture the source of the information for that data element. The field SHALL be a string type, but the value of the field is left to the content producer. The value MAY include, but is not restricted to, a synthetic ID of the asset that sourced the information, another ID of the source, or a description of the source. See the XSD for additional clarity on the "source" attribute.

Each literal data element MAY have a "timestamp" attribute associated with it. If populated, the timestamp attribute indicates when the data was last known to be correct for that element.

6.1. Abstract Elements

6.1.1. ai:asset

INHERITS: None.

DEFINITION: The root element from which all other asset elements derive. This element does not represent any specific type of asset, and therefore should only be used as a base class for other, concrete, asset elements. Any element that claims to be an asset element compliant with this specification SHALL directly or indirectly inherit all of the attributes in this element. The asset element SHALL NOT be used directly in an asset identification instance because it is abstract.

Name	Type	Cnt	Definition
synthetic-id	element-synthetic-id	0-n	Holds the synthetic ID information for the asset.
locations	element - locations	0-1	Holds the location information where the asset resides.
extended-information	element - locations	0-1	Holds extension identifiers for the asset. The content can be any well- formed XML defined in a namespace other than the Asset Identification namespace.
timestamp	literal - dateTime	0-1	The date and time when the information was last known to be correct.

Table 1: ai:asset Properties

6.1.2. ai:it-asset

INHERITS: ai:asset

DEFINITION: An abstract element that extends from the asset element. it-asset is a placeholder element to carry common attributes related to IT assets. For the current iteration of this specification no common attributes have been identified, but future iterations of the specification MAY contain common attributes for IT assets. All asset elements that are describing IT assets SHOULD extend from the it-asset element.

Name	Type	Cnt	Definition
N/A	N/A	N/A	N/A

Table 2: ai:it-asset Properties

6.2. Concrete Asset Elements

The following elements describe the data elements for the asset types defined in this specification.

6.2.1. ai:circuit

INHERITS: ai:it-asset

DEFINITION: Captures identifying information about a circuit.

Name	Type	Cnt	Definition
circuit-name	literal - token	0-1	The name of the circuit being identified.

Table 3: ai:circuit Properties

6.2.2. ai:computing-device

INHERITS: ai:it-asset

DEFINITION: Captures identifying information about a computing device.

Name	Type	Cnt	Definition
distinguished-name	literal - token	0-1	The X.500 distinguished name of the computing device being identified.
cpe	literal - ai:cpe	0-n	The Common Platform Enumeration name for the computing device being identified. This MUST be a hardware CPE. This MUST be a CPE 2.2 URI [CPE22] or CPE 2.3 formatted string [CPE23].
connections	elements - ai:connections	0-1	Information about a network interface on the computing device being identified.
fqdn	literal - token	0-1	The fully-qualified domain name for the computing device being identified.
hostname	literal - token	0-1	The hostname of the computing device.
motherboard-guid	literal - string	0-1	The motherboard globally unique identifier of the computing device.

Table 4: ai:computing-device Properties

6.2.3. ai:data

INHERITS: ai:asset

DEFINITION: A generic element to describe any type of data. Since this element is generic it does not define any of its own properties, but instead relies solely on the properties inherited from asset.

Name	Type	Cnt	Definition
synthetic-id	element-synthetic-id	0-n	Holds the synthetic ID information for the asset.
N/A	N/A	N/A	N/A

Table 5: ai:data Properties

6.2.4. ai:database

INHERITS: ai:it-asset

DEFINITION: Captures identifying information about a database.

Name	Type	Cnt	Definition
instance-name	literal - token	0-1	The name of the database instance.

Table 6: ai:database Properties

6.2.5. ai:network

INHERITS: ai:it-asset

DEFINITION: Captures identifying information about a network.

Name	Type	Cnt	Definition
network-name	literal - normalizedString	0-1	The name of the network being identified.
ip-net-range	element - ip-net-range	0-1 (if ip-net-range, then not cidr)	The starting and ending IP addresses for the range of IP addresses for the network being identified.

cidr	literal - token	0-1 (if cidr, then not ip-net-range)	The Classless Inter-Domain Routing information for the network being identified.
------	-----------------	--	--

Table 7: ai:network Properties

6.2.6. ai:organization

INHERITS: ai:asset

DEFINITION: Captures identifying information about an organization.

Name	Type	Cnt	Definition
xnl:OrganisationNameDetails	element - xnl:OrganisationNameDetails	0-n	The name of the organization being identified. See [xNL] for details on populating this element
email-address	literal - token	0-n	An email address associated with the organization being identified.

telephone-number	literal - token	0-n	A phone number associated with the organization being identified. For a North American number, the number MUST be valid and the format MUST be XXX- XXX-XXXX where X is a digit. For an international number, the number MUST begin with a '+' symbol, followed by 7 to 15 digits. A space MAY be used between digits, as appropriate. For example: +88 888 888 8 (this is following the ITU-T E.123 notation). Regex: ((([2-9][0-8]\d-[2-9]\d{2}-[0-9]{4}) (\+[0-9]?){6,14}[0-9]))
website-url	literal - URL	0-n	A website associated with the organization being identified.

Table 8: ai:organization Properties

6.2.7. ai:person

INHERITS: ai:asset

DEFINITION: Captures identifying information about a person.

Name	Type	Cnt	Definition
xnl:PersonName	element - xnl:PersonName	0-1	The name of the person being identified. The element type is defined in [xNL] and SHALL be used as documented in that specification.
email-address	literal - token	0-n	An email address associated with the person being identified.
telephone-number	literal - token	0-n	A phone number associated with the person being identified. For a North American number, the number MUST be valid and the format MUST be XXX-XXX-XXXX where X is a digit. For an international number, the number MUST begin with a '+' symbol, followed by 7 to 15 digits. A space MAY be used between digits, as appropriate. For example: +88 888 888 8 (this is following the ITU-T E.123 notation). Regex: <code>((([2-9][0-8]\d-[2-9]\d{2}-[0-9]{4}) (\+([0-9]?){6,14}[0-9]))</code>
birthdate	literal - date	0-1	The birth date of the person being identified.

Table 9: ai:person Properties

6.2.8. ai:service

INHERITS: ai:it-asset

DEFINITION: Captures identifying information about a service running on a computing-device.

Name	Type	Cnt	Definition
host	element - host	0-1	The IP address or fully qualified domain name of the host of the service.
port	literal - integer	0-n	The port number that the service is bound to. Restricted to $0 \leq x \leq 65535$
port-range	element - ai:port-range	0-n	The lower and upper bound (inclusive) of the range of ports the service is bound to.
protocol	literal - string	0-1	The protocol used to interact with the service (e.g., HTTP, JMS, SSH, FTP).

Table 10: ai:service Properties

6.2.9. ai:software

INHERITS: ai:it-asset

DEFINITION: Captures identifying information about a class of software or a software instance.

Name	Type	Cnt	Definition
installation-id	literal - token	0-1	Any identifier for a software instance (installation). Use when identifying an instance of software and not just the class of software.
cpe	literal - ai:cpe	0-1	The Common Platform Enumeration name for the class of software being identified. This MUST be a software CPE. This MUST be a CPE 2.2 URI [CPE22] or CPE 2.3 formatted string [CPE23].

license	literal - string	0-n	The license key associated with the software instance (installation). Use when identifying an instance of software and not just the class of software.
---------	------------------------	-----	--

Table 11: ELEMENT NAME PROPERTIES

6.2.10. ai:system

INHERITS: parent

DEFINITION: definition text.

Name	Type	Cnt	Definition
system-name	literal - token	0-n	The name of the system being identified. This property can be replicated as systems may have multiple, or abbreviated, names. All of the names (including acronyms) MAY be captured here.
version	literal - token	0-1	The version of the system being identified.

Table 12: ai:system Properties

6.2.11. ai:website

INHERITS: parent

DEFINITION: definition text.

Name	Type	Cnt	Definition
document-root	literal 1-token	0-1	The absolute path to the document root location of the website on the host.

locale	litera l- token	0-1	The locale of the website represented as an RFC 5646 language, and optionally, region code. Language and region codes SHOULD be in the Internet Assigned Numbers Authority (IANA) Language Subtag Registry [ILSR]. Regex: <code>[a-zA-Z]{2,3}(-([a-zA-Z]{2} [0-9]{3}))?</code>
--------	-----------------------	-----	--

Table 13: ai:website Properties

6.3. Helper Elements

6.3.1. ai:synthetic-id

INHERITS: N/A

DEFINITION: Holds the synthetic identifier information for an asset.

Name	Type	Cnt	Definition
resource	literal - URI	1	A URI for the namespace in which the identifier is governed and unique.
id	literal - token	1	The unique identifier for the asset within the resource namespace.

Table 14: ai:synthetic-id Properties

6.3.2. ai:connections

INHERITS: N/A

DEFINITION: Contains a list of ai:connection elements.

Name	Type	Cnt	Definition
connection	element - ai:connection	1-n	Information about a network interface on the computing device being identified.

Table 15: ai:connections Properties

6.3.3. ai:connection

INHERITS: N/A

DEFINITION: Contains information relevant to a single connection to a network. If multiple IP addresses map to the same MAC address, each ai:connection SHALL represent a single MAC address-IP address pair.

Name	Type	Cnt	Definition
ip-address	literal-token	0-1	The IP address for the connection. IPv4 Regex: ([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5])) IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}
mac-address	literal - ai:mac-address-type	0-1	The Media Access Control address for the network interface. Regex: ([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}
url	literal - URL	0-n	A Universal Resource Locator address for the network interface.
subnet-mask	literal - token	0-1	The subnet mask for the connection. IPv4 Regex: ([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5])) IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}

default-route	literal - token	0-1	<p>The IP address for the default gateway for the connection. IPv4 Regex:</p> <pre>([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))</pre> <p>IPv6 Regex:</p> <pre>([0-9a-fA-F]{1,4}:{1,4}){7}[0-9a-fA-F]{1,4}</pre>
---------------	-----------------	-----	---

Table 16: ELEMENT NAME PROPERTIES

6.3.4. ai:locations

INHERITS: N/A

DEFINITION: Contains a geographic coordinate system point.

Name	Type	Cnt	Definition
location	element - one of: location-point, location-region, location-address (type xal:AddressDetails)	1-n	ai:location is an abstract element that is the root of the substitution group for the elements listed in the type field. Use one of those to describe the location of the asset. xal:AddressDetails is defined in [xAL].

Table 17: ai:locations Properties

6.3.5. ai:location-point

INHERITS: N/A

DEFINITION: Contains a geographic coordinate system point.

Name	Type	Cnt	Definition
latitude	literal - number	1	The latitude of the point represented as a number between -90 and 90. 90 = 90oN, -90 = 90oS. Value constraint: -90 <= x <= 90
longitude	literal - number	1	The longitude of the point represented as a number between -180 and 180. 180 = 180oE, -180 = 180oW. Value constraint: -180 < x <= 180
elevation	literal - number	0-1	The elevation of the point represented in meters above sea level. A negative number would indicate below sea level.
radius	literal - number	0-1	The radius of a horizontal circle centered on the point within which the asset resides. Value constraint: x >= 0

Table 18: ai:location-point Properties

6.3.6. ai:location-region

INHERITS: N/A

DEFINITION: Contains region information.

Name	Type	Cnt	Definition
region-name	literal - normalizedString	1	The name of the region.

Table 19: ai:location-region Properties

6.3.7. ai:ip-range

INHERITS: N/A

DEFINITION: Contains a start and end IP address to create a range.

Name	Type	Cnt	Definition
ip-net-range-start	element - ai:ip-address	1	The start IP address of the range
ip-net-range-end	element - ai:ip-address	1	The end IP address of the range.

Table 20: ELEMENT NAME PROPERTIES

6.3.8. ai:ip-address

INHERITS: N/A

DEFINITION: Contains an IP address.

Name	Type	Cnt	Definition
ip-v4	literal - token	0-1	An IP v4 address. Regex: ([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))\.([0-9] [1-9][0-9] 1([0-9][0-9]) 2([0-4][0-9] 5[0-5]))
ip-v6	literal - token	0-1	An IP v6 address. Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}

Table 21: ai:ip-address Properties

6.3.9. ai:port-range

INHERITS: N/A

DEFINITION: Contains a start and end port number to create a range..

Name	Type	Cnt	Definition
lower-bound	literal - token	1	The lower bound (inclusive) of the range of ports. Restricted to $0 \leq x \leq 65535$
upper-bound	literal - token	1	The lower bound (inclusive) of the range of ports. Restricted to $0 \leq x \leq 65535$. MUST be greater than lower-bound.

Table 22: ai:port-range Properties

6.3.10. ai:host

INHERITS: N/A

DEFINITION: Holds a fully-qualified domain name or an IP address.

Name	Type	Cnt	Definition
fqdn	literal - token	1 (if not ip-address)	The fully-qualified domain name of the host. One of fqdn or ip-address must be specified
ip-address	element - ai:ip-address	1 (if not fqdn)	The IP address of the host. One of fqdn or ip- address must be specified.

Table 23: ai:host Properties

6.3.11. ai:cpe

INHERITS: N/A

DEFINITION: A CPE 2.2 URI [CPE22] or CPE 2.3 formatted string [CPE23].

Name	Type	Cnt	Definition
N/A	N/A	N/A	N/A

Table 24: ai:cpe Properties

6.3.12. Relating Assets to Other Assets

While the assets modeled in Section 6.2, and their related elements, capture the literal values helpful for identifying the respective assets, it is often useful or necessary to define one or more relationships between assets. Those relationships can give additional context to the identifying algorithm in the implementing tool.

The Asset Identification data model allows for explicit relationships to be defined between an asset and one or more other assets. Each relationship is defined as {subject} {predicate} {object}, where {subject} is the asset from which the relationship begins, {predicate} is the relationship type being established, and {object} is one or more other assets. The predicate MUST be a qualified name that refers to a term in a controlled vocabulary. Section 6.4.1 documents the data model to represent assets along with relationships. Section 6.4.2 defines terms in a controlled vocabulary for Asset Identification.

6.3.12.1. Relationship Data Model

Asset Identification defines two elements that can be leveraged by specifications desiring to represent Asset Identification information. The first element, ai:asset-related, SHOULD be leveraged when the implementing specification desires to identify a single asset while demonstrating relationships between that asset and other assets. The second element, ai:assets, SHOULD be leveraged when the implementing specification desires to identify multiple assets while documenting the relationships between those assets and other assets. The two elements are documented in the following tables.

6.3.12.1.1. ai:asset-related

INHERITS: N/A

DEFINITION: Identifies a single asset while capturing the relationships between that asset and other assets.

Name	Type	Cnt	Definition
asset-ref	literal - NCName	1	Contains the ID value of an ai:asset found on this ai:asset-related element. The asset referenced from this property is the primary asset of this element and SHALL be understood to be the asset being identified by this ai:asset-related element.
relationships	element - core:relationships	0-1	Contains the relationships between assets identified in this element.
asset	element - ai:asset	1-n	The assets captured in this element. This includes at minimum the primary asset referenced in asset-ref, as well as any additional assets that the primary asset is related to through a relationship.

Table 25: ai:asset-related Properties

6.3.12.1.2. ai:assets

INHERITS: N/A

DEFINITION: Identifies multiple assets as well as the relationships between the assets.

Name	Type	Cnt	Definition
relationships	element - core:relationships	0-1	Contains the relationships between assets identified in this element
asset	element - ai:asset	1-n	The assets captured in this element.

Table 26: ai:assets Properties

6.3.12.1.3. core:relationships

INHERITS: N/A

DEFINITION: Contains a collection of relationships between the report content and assets, report requests, and other reports.

Name	Type	Cnt	Definition
relationship	element - core:relationship	1-n	Contains a relationship between the subject and object(s) assets.

Table 27: ELEMENT NAME PROPERTIES

6.3.12.1.4. core:relationship

INHERITS: N/A

DEFINITION: Contains a relationship between the subject and object(s) assets.

Name	Type	Cnt	Definition
ref	literal - NCName	1-n	This element MUST identify the object of this relationship by specifying the ID of the asset. Depending on the type of relationship being asserted, there may be additional restrictions on which types of objects may be referenced, but that will be documented with the vocabulary term.
type	literal - QName	1	￼ This element contains the type of relationship that is being specified. The QName MUST refer to a term in a controlled vocabulary. The controlled vocabulary is identified by the namespace URI of the QName, and the term in that controlled vocabulary is specified by the local name of the QName. It is helpful, though not required, that when the namespace URI and local name are concatenated, the resulting URI is dereferenceable and points to a location that defines the term.
scope	literal - token	0-1	Determines how to interpret multiple ref elements in a relationship. If used, this element MUST contain the string "inclusive" or "exclusive". When this element is not provided, its default value is "inclusive". When "inclusive" is specified, this relationship should be understood to exist between the subject asset and the collection of objects identified by the ref elements on this relationship. When "exclusive" is specified, this relationship should be understood to exist between the subject asset and each object asset identified by the ref elements individually.

subject	literal - NCName	1	The property MUST identify the subject of the relationship by specifying the ID of the asset. Depending on the type of relationship being asserted, there may be additional restrictions on which type of asset may be referenced, but that will be documented with the vocabulary term.
---------	------------------------	---	--

Table 28: core:relationship Properties

6.3.12.2. Relationship Types

Defined below are terms in a controlled vocabulary for Asset Identification. It is OPTIONAL that content producers use the terms defined below, but all Asset Identification compliant implementations MUST understand the terms defined in this section. Content producers SHOULD use these terms when possible.

All terms listed in Table 29 exist in the controlled vocabulary identified by <http://scap.nist.gov/specifications/ai/vocabulary/relationships/1.0#>. The definition of each term can also be found at the URL created when concatenating the URL and the term together. The table MUST be interpreted as follows:

- o The "Term" column indicates the local-name of the term being identified.
- o The "Domain" column indicates the exhaustive set of subject types that may be referenced by a relationship of that type. A relationship of that type MUST reference a subject of the type indicated in "Domain" for that relationship.
- o The "Range" column indicates the exhaustive set of object types that may be referenced by a relationship of that type. A relationship of that type MUST reference an object of the type indicated in "Range" for that relationship.
- o The "Description" column contains a prose description of the relationship type. This column may contain requirement words as indicated in [RFC 2119]. Those requirement words MUST be interpreted as described in [RFC 2119] for the relationship.

Term	Domain	Range	Description
hasTerminationDevice	ai:circuit	ai:computing-device	The circuit is terminated by the device
hasServiceProvider	ai:circuit	ai:organization	The circuit is owner/operated by the organization.
hasNetworkTerminationPoint	ai:circuit	ai:network	The circuit ends at the network.
servedBy	ai:database, ai:website	ai:service	The database or website is served up by the service.
hasServiceProvider	ai:service	ai:software	The service is provided by the software.
installedOnDevice	ai:software	ai:computing-device	The software is installed on the computing device.

connectedToNetwork	ai:system	ai:network	The system is connected to the network.
isOwnerOf	ai:person, ai:organization	ai:it-asset	The person or organization owns the IT asset.
isAdministratorOf	ai:person	ai:computing-device, ai:system	The person is the system administrator of the computing device or system.
partOf	ai:person	ai:organization	The person is in some way a part of the organization.
connectedTo	ai:computing-device, ai:system	ai:system	The computing device or system is connected to the system.

Table 29: Controlled Vocabulary Defined for Asset Identification

Content producers that choose to use terms that are not listed in Table 6-29, or to use other terms in addition to those listed in Table 6-29, MAY do so while still remaining compliant to this specification. Content producers SHALL always use terms defined in a controlled vocabulary. The controlled vocabulary SHALL be identified using a URI. Concatenating the controlled vocabulary URI with a term in the vocabulary MAY create a dereferencable URI that points to a definition for that term. This is often accomplished by using an HTTP URL for the controlled vocabulary URI, and ending that URL in

"#" or "/". For instance, <http://scap.nist.gov/specifications/ai/vocabulary/relationships/1.0#installedOnDevice> is a deferenceable link to the definition of "installedOnDevice".

6.3.13. Guidance for Incorporating Asset Identification Elements into Other Data Models

The following guidance applies when incorporating the Asset Identification data model into other data models.

- o If the target data model needs to include a list of assets and the associated relationships between the assets, then the ai:assets element SHOULD be included in the target data model. This will be the case when the target data model will make explicit references to assets in the asset list.
- o If the target data model needs to include a single asset, but other assets are needed to help identify that asset, then the ai:asset-related element SHOULD be included in the target data model. The @asset-ref attribute on ai:asset-related SHALL refer to the specific asset that is being identified.
- o If the target data model needs to include only a single asset, and no relationships are needed to identify that asset, then the element defining the asset MAY be included in the target data model. While this is allowed, it is preferred that the ai:asset-related element be included instead.

7. Asset Identification Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ai="http://scap.nist.gov/schema/asset-identification/1.1"
  xmlns:core="http://scap.nist.gov/schema/reporting-core/1.1"
  xmlns:xal="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xmlns:xnl="urn:oasis:names:tc:ciq:xsdschema:xNL:2.0"
  targetNamespace="http://scap.nist.gov/schema/asset-identification/1.1"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="1.1.1" xmlns:cpe-name="http://cpe.mitre.org/naming/2.0">
  <xs:annotation>
    <xs:appinfo>
      <schema>Asset Identification</schema>
      <author>David Waltermire, Adam Halbardier, John Wunder</author>
      <version>1.1.1</version>
      <date>2012-02-13</date>
    </xs:appinfo>
  </xs:annotation>
  <xs:import namespace="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
    schemaLocation="http://docs.oasis-open.org/election/external/xAL.xsd" />
  <xs:import namespace="urn:oasis:names:tc:ciq:xsdschema:xNL:2.0"
    schemaLocation="http://docs.oasis-open.org/election/external/xNL.xsd" />
  <xs:import namespace="http://scap.nist.gov/schema/reporting-core/1.1"
    schemaLocation="http://scap.nist.gov/schema/reporting-core/1.1/reporting-core
_1.1.0.xsd" />
  <xs:import namespace="http://cpe.mitre.org/naming/2.0"
    schemaLocation="http://scap.nist.gov/schema/cpe/2.3/cpe-naming_2.3.xsd" />
  <!--The following elements are first-order citizens of this model-->
  <xs:element name="asset-related" type="ai:asset-identification-type">
    <xs:key name="assetRelKey">
      <xs:selector xpath="ai:asset" />
      <xs:field xpath="@id" />
    </xs:key>
    <xs:keyref refer="ai:assetRelKey" name="assetRelKeyRef">
      <xs:selector xpath="." />
      <xs:field xpath="@asset-ref" />
    </xs:keyref>
    <xs:keyref refer="ai:assetRelKey" name="assetRelRefSubjKeyRef">
      <xs:selector xpath="core:relationships/core:relationship" />
      <xs:field xpath="@subject" />
    </xs:keyref>
    <xs:keyref refer="ai:assetRelKey" name="assetRelRefObjKeyRef">
      <xs:selector xpath="core:relationships/core:relationship/core:ref" />
      <xs:field xpath="." />
    </xs:keyref>
  </xs:element>

```

```

</xs:element>
<xs:complexType name="asset-identification-type">
  <xs:complexContent>
    <xs:extension base="ai:assets-type">
      <xs:attribute name="asset-ref" type="xs:NCName" use="required"
        />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="assets" type="ai:assets-type">
  <xs:key name="assetsRelKey">
    <xs:selector xpath="ai:asset" />
    <xs:field xpath="@id" />
  </xs:key>
  <xs:keyref refer="ai:assetsRelKey" name="assetsRelRefSubjKeyRef">
    <xs:selector xpath="core:relationships/core:relationship" />
    <xs:field xpath="@subject" />
  </xs:keyref>
  <xs:keyref refer="ai:assetsRelKey" name="assetsRelRefObjKeyRef">
    <xs:selector xpath="core:relationships/core:relationship/core:ref" />
    <xs:field xpath="." />
  </xs:keyref>
</xs:element>
<xs:complexType name="assets-type">
  <xs:complexContent>
    <xs:extension base="core:relationships-container-type">
      <xs:sequence>
        <xs:element name="asset" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="ai:asset" />
            </xs:sequence>
            <xs:attribute name="id" type="xs:NCName" use="required">
              <xs:annotation>
                <xs:documentation>An internal ID to identify this
                  asset.</xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:anyAttribute namespace="##other">
              <xs:annotation>
                <xs:documentation>A placeholder so that content
                  creators can add attributes as
                  desired.</xs:documentation>
              </xs:annotation>
            </xs:anyAttribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

```
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="asset" type="ai:asset-type" abstract="true">
      <xs:annotation>
        <xs:documentation>Holds identifying attributes that are common to
          all assets.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="it-asset" type="ai:it-asset-type" abstract="true"
      substitutionGroup="ai:asset">
      <xs:annotation>
        <xs:documentation>Holds identifying attributes that are common to
          all IT assets</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="circuit" type="ai:circuit-type"
      substitutionGroup="ai:it-asset">
      <xs:annotation>
        <xs:documentation>Holds identifying attributes for a
          circuit.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="computing-device" type="ai:computing-device-type"
      substitutionGroup="ai:it-asset">
      <xs:annotation>
        <xs:documentation>Holds identifying attributes for a computing
          device.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="data" type="ai:data-type"
      substitutionGroup="ai:asset">
      <xs:annotation>
        <xs:documentation>A stub element to represent the identification
          of data. This element can be extended in the future for specific
          types of data.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="database" substitutionGroup="ai:it-asset"
      type="ai:database-type">
      <xs:annotation>
        <xs:documentation>Holds identifying attributes for a
          database.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="network" type="ai:network-type"
      substitutionGroup="ai:it-asset">
      <xs:annotation>
```

```
<xs:documentation>Holds identifying attributes for a
  network.</xs:documentation>
</xs:annotation>
</xs:element>
<!-- This element leverages the OASIS xNL specification which spells it as "org
anisation" -->
<xs:element name="organization" type="ai:organization-type"
  substitutionGroup="ai:asset">
  <xs:annotation>
    <xs:documentation>Holds identifying attributes for an
      organization.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="person" type="ai:person-type"
  substitutionGroup="ai:asset">
  <xs:annotation>
    <xs:documentation>Holds identifying attributes for a
      person.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="service" substitutionGroup="ai:it-asset"
  type="ai:service-type">
  <xs:annotation>
    <xs:documentation>Holds identifying attributes for a
      service.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="software" type="ai:software-type"
  substitutionGroup="ai:it-asset">
  <xs:annotation>
    <xs:documentation>Holds identifying attributes for a software
      installation</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="system" type="ai:system-type"
  substitutionGroup="ai:it-asset">
  <xs:annotation>
    <xs:documentation>Holds identifying attributes for a
      system.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="website" substitutionGroup="ai:it-asset"
  type="ai:website-type">
  <xs:annotation>
    <xs:documentation>Holds identifying attributes for a
      website.</xs:documentation>
  </xs:annotation>
</xs:element>
<!--The following types correspond directly to the elements above-->
```

```
<xs:complexType name="asset-type" abstract="true">
  <xs:sequence>
    <xs:element ref="ai:synthetic-id" minOccurs="0"
      maxOccurs="unbounded" />
    <xs:element ref="ai:locations" minOccurs="0" />
    <xs:element name="extended-information" minOccurs="0">
      <xs:annotation>
        <xs:documentation>This is a container to hold any additional
          identifying information for an asset, as specified by the
          content creator.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##other" processContents="lax"
            maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute ref="ai:timestamp" />
</xs:complexType>
<xs:complexType name="it-asset-type" abstract="true">
  <xs:complexContent>
    <xs:extension base="ai:asset-type" />
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="circuit-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
        <xs:element name="circuit-name" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The common name for the
              circuit.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:token">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
</xs:complexType>
<xs:complexType name="computing-device-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
        <xs:element name="distinguished-name" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The full X.500 distinguished name for
              the device.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:token">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element ref="ai:cpe" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>The hardware CPE name for the device
              (CPE 2.2 URI or CPE 2.3 Formatted
              String).</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="connections" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The IP network interface connections
              that exist for the device (regardless of if the network
              interface is connected to a network or
              not).</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="connection"
                type="ai:network-interface-type" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>An IP network interface
                    connection.</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element minOccurs="0" ref="ai:fqdn" />
        <xs:element name="hostname" minOccurs="0">
```



```
<xs:annotation>
  <xs:documentation>The hostname of the computing
    device.</xs:documentation>
</xs:annotation>
<xs:complexType>
  <xs:simpleContent>
    <xs:extension base="ai:hostname-type">
      <xs:attribute ref="ai:source" />
      <xs:attribute ref="ai:timestamp" />
      <xs:anyAttribute namespace="##other" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="motherboard-guid">
  <xs:annotation>
    <xs:documentation>The motherboard globally unique
      identifier of the computing device.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="data-type">
  <xs:complexContent>
    <xs:extension base="ai:asset-type" />
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="database-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
        <xs:element minOccurs="0" name="instance-name">
          <xs:annotation>
            <xs:documentation>The name of the database instance being
              identified.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<xs:simpleContent>
  <xs:extension base="xs:token">
    <xs:attribute ref="ai:source" />
    <xs:attribute ref="ai:timestamp" />
    <xs:anyAttribute namespace="##other" />
  </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="network-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
        <xs:element name="network-name" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The name of the network as commonly
              referred to.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:normalizedString">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:choice minOccurs="0">
          <xs:element name="ip-net-range">
            <xs:annotation>
              <xs:documentation>The start and end IP addresses to
                indicate the range of IP addresses covered by this
                network.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ip-net-range-start"
                  type="ai:ip-address-type">
                  <xs:annotation>
                    <xs:documentation>The starting IP address in the
                      network.</xs:documentation>
                  </xs:annotation>
                </xs:element>
```

```
<xs:element name="ip-net-range-end"
  type="ai:ip-address-type">
  <xs:annotation>
    <xs:documentation>The ending IP address in the
      network.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="cidr">
  <xs:annotation>
    <xs:documentation>The classless inter-domain routing
      notation for the network.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="ai:cidr-type">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:choice>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="organization-type">
  <xs:complexContent>
    <xs:extension base="ai:asset-type">
      <xs:sequence>
        <xs:element ref="xnl:OrganisationNameDetails" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="ai:email-address" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="ai:telephone-number" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element ref="ai:website-url" minOccurs="0"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="person-type">
  <xs:complexContent>
```

```
<xs:extension base="ai:asset-type">
  <xs:sequence>
    <xs:element ref="xnl:PersonName" minOccurs="0" />
    <xs:element ref="ai:email-address" minOccurs="0"
      maxOccurs="unbounded" />
    <xs:element ref="ai:telephone-number" minOccurs="0"
      maxOccurs="unbounded" />
    <xs:element name="birthdate" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The birthdate of the
          person.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:date">
            <xs:attribute ref="ai:source" />
            <xs:attribute ref="ai:timestamp" />
            <xs:anyAttribute namespace="##other" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="service-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
        <xs:element minOccurs="0" name="host">
          <xs:annotation>
            <xs:documentation>The hostname or IP address where the
              service is hosted.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:choice>
              <xs:element ref="ai:fqdn" />
              <xs:element ref="ai:ip-address" />
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <xs:element name="port" maxOccurs="unbounded" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The port to which the service is
              bound.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
```

```
<xs:simpleContent>
  <xs:extension base="ai:port-type">
    <xs:attribute ref="ai:source" />
    <xs:attribute ref="ai:timestamp" />
    <xs:anyAttribute namespace="##other" />
  </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element name="port-range" maxOccurs="unbounded"
  minOccurs="0">
  <xs:annotation>
    <xs:documentation>The inclusive port range to which the
      service is bound.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="lower-bound" type="ai:port-type"
      use="required" />
    <xs:attribute name="upper-bound" type="ai:port-type"
      use="required" />
    <xs:attribute ref="ai:source" />
    <xs:attribute ref="ai:timestamp" />
    <xs:anyAttribute namespace="##other" />
  </xs:complexType>
</xs:element>
<xs:element name="protocol" minOccurs="0">
  <xs:annotation>
    <xs:documentation>The protocol used to interact with the
      service.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="software-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
```

```
<xs:element name="installation-id" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Any identifier for the software instance
      (installation)</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:token">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element ref="ai:cpe" minOccurs="0">
  <xs:annotation>
    <xs:documentation>The CPE name for the software (CPE 2.2
      URI or CPE 2.3 Formatted String).</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="license" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>The license key for the
      software.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="system-type">
  <xs:complexContent>
    <xs:extension base="ai:it-asset-type">
      <xs:sequence>
        <xs:element name="system-name" minOccurs="0"
          maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>The name of the system. It is possible
```

```
        that a system have multiple names, or even abbreviated
        names. Each one of those names may be captured
        here.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="version" minOccurs="0">
    <xs:annotation>
        <xs:documentation>The version of the
            system.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="website-type">
    <xs:complexContent>
        <xs:extension base="ai:it-asset-type">
            <xs:sequence>
                <xs:element name="document-root" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>The absolute path to the document root
                            location of the website on the host.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:token">
                                <xs:attribute ref="ai:source" />
                                <xs:attribute ref="ai:timestamp" />
                                <xs:anyAttribute namespace="##other" />
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

```

        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="locale" minOccurs="0">
    <xs:annotation>
      <xs:documentation>The locale of the website represented as
        an RFC 5646 language, and optionally, region
        code.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ai:locale-type">
          <xs:attribute ref="ai:source" />
          <xs:attribute ref="ai:timestamp" />
          <xs:anyAttribute namespace="##other" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<!--The follow elements support the first-order citizen elements defined above-->
<xs:element name="cpe">
  <xs:annotation>
    <xs:documentation>A Common Platform Enumeration (CPE) name (CPE
      2.2 URI or CPE 2.3 Formatted String).</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="ai:cpe-type">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="synthetic-id">
  <xs:annotation>
    <xs:documentation>Holds the synthetic ID for the
      asset</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="resource" type="xs:anyURI" use="required">
      <xs:annotation>

```



```
        <xs:documentation>The namespace governing this synthetic
            ID.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="id" type="xs:token" use="required">
    <xs:annotation>
        <xs:documentation>The ID of the asset within the resource
            namespace.</xs:documentation>
    </xs:annotation>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="email-address">
    <xs:annotation>
        <xs:documentation>An email address</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="fqdn">
    <xs:annotation>
        <xs:documentation>The fully qualified domain name for the
            object.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="ip-address" type="ai:ip-address-type">
    <xs:annotation>
        <xs:documentation>An IP address</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="location" abstract="true" />
<xs:element name="location-address" type="xal:AddressDetails"
```

```
substitutionGroup="ai:location">
  <xs:annotation>
    <xs:documentation>The address where an asset is
      located.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="location-point" substitutionGroup="ai:location">
  <xs:annotation>
    <xs:documentation>The geographic point where an asset is
      located.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="latitude" use="required">
      <xs:annotation>
        <xs:documentation>The latitude of the asset, defined between
          -90 (90 degrees South, inclusive) and 90 (90 degrees North,
            inclusive).</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:double">
          <xs:minInclusive value="-90" />
          <xs:maxInclusive value="90" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="longitude" use="required">
      <xs:annotation>
        <xs:documentation>The longitude of the asset, defined between
          -180 (180 degrees West, exclusive) and 180 (180 degrees
            East, inclusive).</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:double">
          <xs:minExclusive value="-180" />
          <xs:maxInclusive value="180" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="elevation" type="xs:double">
      <xs:annotation>
        <xs:documentation>The elevation of the asset, specified in
          meters from sea level.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="radius">
      <xs:annotation>
        <xs:documentation>The radius of a horizontal circle centered
          on the point within which the asset
```

```
        resides.</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:double">
          <xs:minInclusive value="0" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="ai:source" />
    <xs:attribute ref="ai:timestamp" />
    <xs:anyAttribute namespace="##other" />
  </xs:complexType>
</xs:element>
<xs:element name="location-region" substitutionGroup="ai:location">
  <xs:annotation>
    <xs:documentation>The region where an asset is
      located.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:normalizedString">
        <xs:attribute ref="ai:source" />
        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="locations">
  <xs:annotation>
    <xs:documentation>One or more locations where this asset
      resides</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ai:location" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>The base for a substitution group for
            elements that contain location
            information.</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="served-by" type="ai:service-type">
  <xs:annotation>
    <xs:documentation>The service that is serving up the
```

```

        asset.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="telephone-number">
    <xs:annotation>
        <xs:documentation>The telephone number. For a North American
            number, the number must be valid and the format must be
            XXX-XXX-XXXX where X is a digit. For an international number,
            the number must begin with a '+' symbol, followed by 7 to 15
            digits. A space may be used between digits, as appropriate. For
            example: +88 888 888 8 (this is following the ITU-T E.123
            notation).</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="ai:telephone-number-type">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="website-url">
    <xs:annotation>
        <xs:documentation>The URL to the website.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<!--The following types support the supporting elements above-->
<xs:attribute name="source" type="xs:string">
    <xs:annotation>
        <xs:documentation>Contains the source of the information. The
            value of this field is left open to the content producer, but
            MAY include a synthetic ID of the asset which sourced the
            information, another ID of the source, or a description of the
            source.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="timestamp" type="xs:dateTime">

```

```
<xs:annotation>
  <xs:documentation>Indicates when the data was last known to be
    correct.</xs:documentation>
</xs:annotation>
</xs:attribute>
<xs:complexType name="ip-address-type">
  <xs:sequence>
    <xs:element name="ip-v4" minOccurs="0">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="ai:ipv4-type">
            <xs:attribute ref="ai:source" />
            <xs:attribute ref="ai:timestamp" />
            <xs:anyAttribute namespace="##other" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="ip-v6" minOccurs="0">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="ai:ipv6-type">
            <xs:attribute ref="ai:source" />
            <xs:attribute ref="ai:timestamp" />
            <xs:anyAttribute namespace="##other" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="network-interface-type">
  <xs:sequence>
    <xs:element ref="ai:ip-address" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The IP address for this network
          interface.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="mac-address" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The MAC address associated with this network
          interface.</xs:documentation>
      </xs:annotation>
    </xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ai:mac-address-type">
          <xs:attribute ref="ai:source" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:sequence>
</xs:complexType>
```

```

        <xs:attribute ref="ai:timestamp" />
        <xs:anyAttribute namespace="##other" />
    </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element name="url" minOccurs="0" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>A URL in a relevant DNS server for this IP
            address.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute ref="ai:source" />
                <xs:attribute ref="ai:timestamp" />
                <xs:anyAttribute namespace="##other" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="subnet-mask" type="ai:ip-address-type"
    minOccurs="0">
    <xs:annotation>
        <xs:documentation>The subnet mask address for this network
            interface.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="default-route" type="ai:ip-address-type"
    minOccurs="0">
    <xs:annotation>
        <xs:documentation>The IP address for the default gateway of
            this connection.</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="cidr-type">
    <xs:restriction base="xs:token">
        <xs:pattern
            value="([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9]
            |[0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-
            4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))/([0-9]|[1
            -2][0-9]|3[0-2])"
            />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="cpe-type">
    <xs:union memberTypes="cpe-name:cpe22Type cpe-name:cpe23Type" />
</xs:simpleType>
<xs:simpleType name="hostname-type">

```

```

    <xs:restriction base="xs:token">
      <xs:pattern value="[\w\-\-]+(\.([\w\-\-]+){0,})" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipv4-type">
    <xs:restriction base="xs:token">
      <xs:pattern
        value="([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9]
        |[0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-
        4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))"
        />
      </xs:restriction>
    </xs:simpleType>
  <xs:simpleType name="ipv6-type">
    <xs:restriction base="xs:token">
      <xs:pattern value="([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="locale-type">
    <xs:restriction base="xs:token">
      <xs:pattern value="[a-zA-Z]{2,3}(-([a-zA-Z]{2}|[0-9]{3}))?" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="mac-address-type">
    <xs:restriction base="xs:token">
      <xs:pattern value="([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="port-type">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="65535" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="telephone-number-type">
    <xs:restriction base="xs:token">
      <xs:pattern
        value="((([2-9][0-8]\d-[2-9]\d{2}-[0-9]{4})|(\+([0-9] ?){6,14}[0-9]))"
        />
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>

```

8. IANA Considerations

This document uses URNs to describe an XML namespace and schema conforming to a registry mechanism described in [RFC3688].

Registration for the IODEF namespace:

- o URI: urn:ietf:params:xml:ns:asset-identification-1.0
- o Registrant Contact: See the first author of the "Author's Address" section of this document.
- o XML: None. Namespace URIs do not represent an XML specification.
- o URI: urn:ietf:params:xml:schema:asset-identification-1.0
- o Registrant Contact: see the first author of the "Author's Address" section of this document.
- o XML: See the "Asset Identification Schema" in ietf-montville-sacm-asset-identification-00.

9. Security Considerations

As a data format, Asset Identification does not intrinsically have any real security concerns - at least none are known at this time. However, as a data format designed to be stored and transmitted between entities within an enterprise, the fact of the matter is that it SHOULD be used within a properly secured environment. Over time, a significant amount of information valuable to attackers can be gleaned from Asset Identification information (i.e. derived topology, asset purpose and criticality, to name just two). Therefore, it is recommended that use of Asset Identification expressions be performed in environments providing communication security mechanisms supplying the properties of confidentiality, data integrity, and non-repudiation. For example, transmission over a properly configured TLS connection would be better than being sent in the clear with no protection.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.
- [RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [CPE23] "NIST Interagency Report 7695, Common Platform Enumeration: Naming Specification Version 2.3", April 2011.
- [ILSR] "IANA Language Subtag Registry".
- [XAL] "Organization for the Advancement of Structured Information Standards (OASIS) Extensible Address Language (xAL) Version 2.0", July 2002.
- [XNL] "Organization for the Advancement of Structured Information Standards (OASIS) Extensible Naming Language (xNL) Version 2.0", May 2002.
- [IR7693] Wunder, J., Halbardier, A., and D. Waltermire, "Specification for Asset Identification 1.1", June 2011.

10.2. Informative References

- [CPE22] "Common Platform Enumeration (CPE) Specification Version 2.2", March 2009.

Appendix A. Acknowledgements

Special thanks go to John Wunder, Adam Halbardier, and David Waltermire, and those others who participated in the definition of the Asset Identification Specification version 1.1 [IR7693]

Appendix B. Use Cases

The following use cases describe some common asset management processes that rely on the ability to uniquely identify assets. The asset identification specification was developed primarily in order to support these use cases, although the specification may be useful for other processes or uses.

B.1. Correlation of Sensed Data

Sensor data is not limited to an automated process: user surveys, manually entered information, and other data may also be correlated using this Asset Identification specification. The data must be correlated both with other manually collected data and with data collected by automated sensors, in order to build a complete representation of all known data about an asset.

Consistent asset identification allows data to be correlated regardless of:

- o Collection timeframe
- o Data type (vulnerability scan vs. user survey)
- o Manual or automated process
- o Data format

In this case, the goal of asset identification is to provide as much identifying information as possible about an asset in order to ensure the greatest probability of matching asset data from several different sources. Constraining which data is provided merely reduces the possibility of a match.

B.2. Federation of Asset Databases

While many smaller organizations may only have a single asset database, larger organizations with many asset databases may wish to share information about assets among them. This includes:

- o Peer to peer relationships, where asset data is replicated and fused between several asset databases
- o Hierarchical relationships, where an asset database is aggregated from several lower-level databases into fewer higher-level databases

In both use cases, asset identification facilitates detection of

duplicate asset representations, correlation of asset data across the databases, and direct queries for asset data among tools.

In this use case, asset databases may wish to use a smaller set of data, or even a single identifier, in order to properly federate the asset data. For example, use of the motherboard GUID or single synthetic ID, such as an asset tag, may be sufficient to allow asset databases to exchange information about assets.

B.3. Directly Targeted Remediation Actions

While assessment and sensor data may be collected from all assets in a particular organization environment without specifically identifying a particular asset identifier, remediation actions require a more granular identification process that directly targets the asset using identification data. This ensures that unintended side effects are avoided and the intended remediation action is able to be completed successfully.

A single agent identifier or motherboard GUID allows those triggering remediation actions to specify exactly which assets should have the remediation applied and allows the tools to unambiguously identify those assets in the remediation control language.

B.4. Management of Asset Data

Outside of the collection of sensor data and federation of asset databases, asset data may be used in a variety of management processes. This includes both further processing of asset data, such as aggregation for the purposes of metrics collection, and display to an end user. Both of these uses require asset identification be present to ensure all systems are able to accurately represent the correct assets. For purposes of aggregation, for example, asset identification may be used to request detailed data about outliers from the sensors that collected the data.

Appendix C. Extending the Asset Identification Specification

Although the core Asset Identification specification should satisfy most users, some organizations may find that there are weaknesses or missing elements in the specification. In these cases, the organization may extend the Asset Identification specification by extending the XML schema that defines the data model. These extensions should be published as an XML schema to ensure that both schema extensions and instance documents are valid.

C.1. Additional Asset Types

Some organizations may wish to identify assets that are not contained in the valid asset list as defined in Section 5.3. To do this, additional XML elements can be defined that extend (using the XSD extension mechanism) an appropriate asset type (best practices entail using the most specific asset type that captures the necessary elements). Additional identification fields can be defined in the schema and additional relationships can be defined by creating new relationship types in a non-core controlled vocabulary.

C.2. Additional Literal Identifiers for Existing Asset Types

Additional literal identifiers for existing asset types may be added by creating schema elements in the extended-information element. Although there is no technical restriction of these fields to literal identifiers (i.e. information that is used to identify the asset), placing information in an asset element that does not help identify the asset is counter to the purpose of this specification.

C.3. Additional Relationships

Organizations may also wish to identify existing or new asset types using relationships that are not defined in the core specification. This may be accomplished by defining and publishing a relationship vocabulary in a separate namespace than the core Asset Identification relationship vocabulary.

C.4. Additional Properties on Existing Data Elements

The XML schema also provides extension points on most XML elements to allow for tracking metadata about Asset Identification information, such as classification level, sensitivity, or other organization-specific data. As long as Asset Identification elements are able to validate against the core schema and conform to the requirements of this specification, these extensions and the Asset Identification elements are valid.

Author's Address

Adam W. Montville (editor)
Tripwire, Inc.
101 SW Main Street, Suite 1500
Portland, Oregon 98662
USA

Email: amontville@tripwire.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 17, 2012

D. Waltermire, Ed.
NIST
May 16, 2012

Automated XML Content Data Exchange and Management
draft-waltermire-content-repository-00

Abstract

TBD...

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 November 17, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	7
1.2. Terms	7
1.2.1. Content	7
1.2.2. Security Automation Content	7
1.2.3. Content Producer	7
1.2.4. Content Consumer	7
1.2.5. Content Bundle	7
2. Key Concepts	7
2.1. The Content Metadata Model	7
2.2. Content federation	8
3. IANA Considerations	8
4. Security Considerations	8
5. References	8
5.1. Normative References	8
5.2. Informative References	9
Appendix A. Additional Stuff	9
Author's Address	9

1. Introduction

Data-driven programming is a common paradigm in software engineering. When using this approach, a program is developed to process a series of data statements that describe the sequence of actions to be taken. These data statements, often referred to as content, provide the user with a dynamic degree of control over the function of the software. In many cases, this approach can lead to a proliferation of content. Without adequate content management and distribution capabilities, use of content can become impractical.

It is common practice today to format content using the Extensible Markup Language XML . While many content management solutions exist today, few are designed to support the management and distribution of XML-based content. Current solutions largely focus on exploiting the raw XML syntax or a specific data model. Some solutions, such as XML databases, expose the raw syntax of XML for querying using techniques like XQuery. Other solutions utilize specialized database schema designed to support one or more specific data models represented in XML using XML Schema . These solutions are often brittle, inflexible to revisions of the underlying data models and do not adequately represent the logical information components used within data-driven programs.

XML-based data-driven content is produced by many organizations in a range of formats, covering many different information domains. Where content repositories exist to support this content, they often operate independently and vary in the data models and capabilities they support. Rarely do these repositories interact and if they do it is through proprietary interfaces. Content consumers often have to manually download the content they want to use with their tools. In many cases they may want to customize this content for local use and must contend with managing updates to the content manually.

One example of where data-driven programming is used is in the IT Security Automation community. Standardized security automation content is used to provide the instructions necessary for security tools to examine a computer's state to evaluate and report on the degree of compliance to configuration policies, to detect the presence of vulnerabilities, and to verify the installation state of patches. Other tools use data-driven content to collect and correlate digital events or to aggregate security information. Much of the focus in the security automation community has been on defining the standards and schemas for expressing security-related data in XML. Standardizing the methods for retrieval and exchange of security automation content has not been a primary area of focus.

The content management challenges introduced by diverse data models,

decentralized production and use of content, and the proprietary nature of content repositories today create a need to define common content exchange requirements and mechanisms that will complement the content specifications and XML schemas.

The following challenges are addressed by this specification:

Distribution - In the absence of a standardized, automated distribution mechanism, content producers have no way to notify content consumers when new or updated content is available. Content consumers must manually import content at the point of use. This specification defines an automated notification mechanism that can be used to indicate to content consumers when new or updated content is available. The specification also defines the technical mechanisms used to exchange content between repositories, providing a standardized delivery mechanism to make remotely published content available at the point of use.

Reuse

Without a standardized method to search, retrieve and utilize existing content, both content consumers and producers have a tendency to recreate content. This duplication often causes content to become static or stale, introduces errors, and reduces the efficiency for developing content. In support of making content more reusable, this specification provides mechanisms for querying content so that it can be searched and gathered from many content providers. This allows organizations that are developing content to leverage, extend, and customize existing content from a variety of sources. This specification also defines a stable method of identifying blocks of externally provided content enabling content to be remotely referenced. This approach supports reuse and reduces the need for manual duplication across repositories.

Interoperability

Content repositories may require proprietary clients or tools to access their content. This hampers the ability for a content consumer to retrieve content from a variety of content sources using a single tool implementation. This specification standardizes the methods used to publish to and retrieve content from a content repository enabling standardized clients to be developed.

Access to content repositories may be restricted or require the use of various standard or proprietary communication protocols (e.g. HTTP, FTP). Content is often packaged using various

file formats and compression algorithms, such as Zip, CAB or GZIP. Variation in these approaches hampers interoperability. This specification standardizes the communication protocol and distribution formats used promoting interoperability.

Content packaging

XML-based content is exchanged as XML documents, also called instances. This document centric view of information does not align well with how humans use information. Humans are more comfortable working with logical objects that represent a concept (e.g. rule, assessment check, logical construct) verses XML syntax. While XML Schema enables these concepts to be modeled, XML is still represented as a collection of elements and attributes. This specification defines a metamodel that identifies the logical objects that are represented in XML-based content and their boundaries within the XML model enabling content repositories to use the conceptual view of the content.

This technique enables XML instances to be treated as containers of conceptual constructs. These conceptual constructs can be exchanged individually and can be composed into new documents dynamically based on metadata rules. This specification will provide a methodology for gathering and packaging content based on the needs or interest of the content consumer using a metadata approach.

Integrity

Content consumers need assurances that the content that has been received has not been modified during the exchange process. This specification defines the use of automated mechanisms for verifying the integrity of exchanged content.

Confidentiality

In some scenarios, it is necessary to secure the exchange of content or restrict access to specific content. This specification will detail mechanisms for securing repository-to-repository and client-to-repository communications. Additionally this specification will specify authorization mechanisms that enable restricted access to content if needed.

Content Version Management

The content managed by content repositories may often undergo revision. When revisions occur, it is important to be able to

query specific revision to maintain the integrity of content bundles. This specification provides a query method that enables either a specific revision or the latest revision to be retrieved. This approach also enables remote references to include a content identifier and a specific revision.

Model Revision Management

Content repositories are often based on a specific data specification revision. When using this approach, updating content repository software to support specification revisions may require costly, time-consuming effort. Organizations maintaining content repositories may be reluctant to adopt new revisions or support old revisions due to this burden. This makes it difficult for a tool to use content based on an older or newer model revision. This specification defines properties within the metadata model to indicate where content is backwards and forwards compatible. These properties are then used to enable content to be provided based on the required model revision or to drive proper error handling where content is incompatible.

For example the Open Vulnerability and Assessment Language (OVAL) versions content based on the major and minor revision of the OVAL XML schema. A repository containing OVAL content may have content ranging from OVAL 5.3 to 5.10. The difference in model version, while minor, could negatively impact a security tool's ability to properly process content that is outside of its expected range. This could cause tool errors or unexpected results to be produced. By using the model revision properties in the metamodel, the effective model revision of content returned from a content repository may be calculated based on the maximum schema revision used. Alternately, substitute content may be provided that supports a specific maximum schema revision provided in the query.

By addressing these challenges, content producers will be able to effectively manage and share content they produce, and content consumers will be able to effectively use content provided by many different providers. By defining communication interfaces that can leverage existing communication protocols, we can begin to automate content distribution among disparate systems and make content more readily available. By defining a federated data model, we can establish rules and relationships of data types which allow for flexible content management with support for dynamic methods for collecting and bundling content for consumers.

Sections [...] of this document focus on:

TBD

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].

1.2. Terms

1.2.1. Content

1.2.2. Security Automation Content

1.2.3. Content Producer

1.2.4. Content Consumer

1.2.5. Content Bundle

2. Key Concepts

This section provides a high-level overview of key concepts introduced in this specification. The first concept subsection describes a content metamodel that provides a needed level of abstraction over XML-based data models. The second subsection describes the federated content architectural approach defined within this specification. Through the use of these concepts, a robust, general purpose, distributed content management system is possible that supports automated content exchange between content consumers and producers.

2.1. The Content Metadata Model

In order to create a generalized approach to XML-based content management it is necessary to generalize how XML-based data is processed by the content system. A variety of XML schema languages are used to define the syntax used to express a data model in XML. While these languages provide rules to constrain XML instance data, they do not adequately describe the information objects that exist within the model or the relationships between information objects. An information object is a block of XML data that represents a specific concept such as policy definition, a configuration setting or a scanning rule. Relationships represent cross references or links between information objects. Information objects and relationships are concepts that humans use to conceptualize the data model primitives that exist within content. In order for a content

management approach to be successful, a mechanism is needed that bridges the gap between the XML syntax understood by machines and the conceptual primitives that humans understand. The content metadata model provides this bridge.

Within the content metamodel, an information object is represented as an entity definition.

Complete this section...

2.2. Content federation

Complete this section...

Discuss

Use of namespaces within content identifiers for repository lookup using DNS SRV records. Discuss using external namespaces for other cases.

Discuss authoritative content repositories vs. caching repository content.

Discuss using an architectural model similar to DNS for content repositories (e.g. local, forwarding, caching).

3. IANA Considerations

This memo includes no request to IANA.

4. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

5.2. Informative References

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

Appendix A. Additional Stuff

This becomes an Appendix if needed.

Author's Address

David Waltermire (editor)
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Phone:

Email: david.waltermire@nist.gov

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 11, 2013

D. Waltermire, Ed.
NIST
A. Montville
TW
September 7, 2012

Analysis of Security Automation and Continuous Monitoring (SACM) Use
Cases

draft-waltermire-sacm-use-cases-02

Abstract

This document identifies foundational use cases, derived functional capabilities and requirements, architectural components, and the supporting standards needed to define an interoperable, automation\infrastructure required to support timely, accurate and actionable situational awareness over an organization's IT systems. Automation tools implementing a continuous monitoring approach will utilize this infrastructure together with existing and emerging event, incident and network management standards to provide visibility into the state of assets, user activities and network \behavior. Stakeholders will be able to use these tools to aggregate and analyze relevant security and operational data to understand the organizations security posture, quantify business risk, and make informed decisions that support organizational objectives while protecting critical information. Organizations will be able to use these tools to augment and automate information sharing activities to collaborate with partners to identify and mitigate threats. Other automation tools will be able to integrate with these capabilities to enforce policies based on human decisions to harden systems, prevent misuse and reduce the overall attack surface.

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, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	6
1.1. Requirements Language	6
2. Key Concepts	7
3. Use Cases	9
3.1. UC1: System State Assessment	9
3.1.1. Goal	9
3.1.2. Main Success Scenario	9
3.1.3. Extensions	9
3.2. UC2: Enforcement of Acceptable State	9
3.2.1. Goal	9
3.2.2. Main Success Scenario	9
3.2.3. Extensions	10
3.3. UC3: Security Control Verification and Monitoring	10
3.3.1. Goal	10
3.3.2. Main Success Scenario	10
3.3.3. Extensions	10
4. Functional Capabilities	10
4.1. Capabilities Supporting UC1	11
4.1.1. Asset Management	11
4.1.2. Data Collection	12
4.1.2.1. Security Configuration Management	12
4.1.2.2. Vulnerability Management	12
4.1.3. Assessment Result Analysis	13
4.1.4. Content Management	13
4.2. Capabilities Supporting UC2	14
4.2.1. Assessment Query and Transport	14
4.2.2. Acceptable State Enforcement	14
4.3. Capabilities Supporting UC3	14
4.3.1. Tasking and Scheduling	14
4.3.2. Data Aggregation and Reporting	15
5. Functional Components	16
5.1. Asset Management	16
5.1.1. Discovery	16
5.1.2. Characterization	16
5.1.2.1. Logical	16
5.1.2.2. Security	16
5.1.3. Asset Identification	16
5.2. Security Configuration Management	16
5.2.1. Configuration Assessment	16
5.2.1.1. Non-technical Assessment	16
5.2.1.2. Technical Assessment	17
5.3. Vulnerability Management	17
5.3.1. Non-technical Vulnerability Assessment	17
5.3.2. Technical Vulnerability Assessment	17
5.4. Content Management	17
5.4.1. Control Frameworks	17

5.4.2.	Configuration Standards	17
5.4.3.	Scoring Models	17
5.4.4.	Vulnerability Information	17
5.4.5.	Patch Information	17
5.4.6.	Asset Information	17
5.5.	Assessment Result Analysis	17
5.5.1.	Comparing Actual to Expected State	17
5.5.2.	Scoring Comparison Results	17
5.5.3.	Relating Comparison Results to Requirements	17
5.5.4.	Relating Requirements to Control Frameworks	17
5.6.	Tasking and Scheduling	17
5.6.1.	Selection of Assessment Criteria	18
5.6.2.	Defining In-scope Assets	18
5.6.3.	Defining Periodic Assessments	18
5.6.4.	Defining Assessment Triggers	18
5.7.	Data Aggregation and Reporting	18
5.7.1.	By Asset Characterization	18
5.7.2.	By Assessment Criteria	18
5.7.3.	By Control Framework	18
5.7.4.	By Benchmark	18
5.7.5.	By Ad Hoc/Extended Properties	18
6.	Data Exchange Models and Communications Protocols	18
6.1.	Data Exchange Models	19
6.1.1.	Control Expression	19
6.1.1.1.	Technical Control Expression	19
6.1.1.2.	Non-technical Control Expression	19
6.1.2.	Control Frameworks	19
6.1.2.1.	Logical Expression and Syntactic Binding(s)	19
6.1.2.2.	Relationships	19
6.1.2.3.	Substantiation (Control Requirement)	19
6.1.2.4.	Reporting	19
6.1.3.	Asset Expressions	19
6.1.3.1.	Asset Identification	19
6.1.3.2.	Asset Classification (Type)	19
6.1.3.3.	Asset Attributes	20
6.1.3.4.	Information Expression (non-identifying)	20
6.1.3.5.	Reporting	20
6.1.4.	Benchmark/Checklist Expression	20
6.1.4.1.	Logical Expression and Bindings	20
6.1.4.2.	Checking Systems	20
6.1.4.3.	Results and Scoring	20
6.1.4.4.	Reporting	20
6.1.5.	Check Language	20
6.1.5.1.	Logical Expression and Syntactic Binding(s)	20
6.1.5.2.	Reporting	20
6.1.6.	Targeting Expression	20
6.1.6.1.	Information Owner	20
6.1.6.2.	System Owner	20

6.1.6.3. Assessor	20
6.1.6.4. Computing Device	20
6.1.6.5. Targeting Extensibility	20
6.2. Communication Protocols	21
6.2.1. Asset Management Interface	21
7. IANA Considerations	21
8. Security Considerations	21
9. Acknowledgements	21
10. References	21
10.1. Normative References	21
10.2. Informative References	21
Appendix A. Additional Stuff	22
Authors' Addresses	22

1. Introduction

This document addresses foundational use cases in security automation. These use cases may be considered when establishing a charter for the Security Automation and Continuous Monitoring (SACM) working group within the IETF. This working group will address a many of the standards needed to define an interoperable, automation infrastructure required to support timely, accurate and actionable situational awareness over an organization's IT systems. This document enumerates use cases and breaks down related concepts that cross many IT security information domains.

Sections Section 2, Section 3, Section 4, and Section 5 of this document respectively focus on:

- Defining the key concepts and terminology used within the document providing a common frame of reference;

- Identifying foundational use cases that represent classes of stakeholders, goals, and usage scenarios;

- A set of derived functional capabilities and associated requirements that are needed to support the use cases;

- A break down of architectural components that address one or more functional capabilities that can be used in various combinations to support the use cases

The concepts identified in this document provide a foundation for creating interoperable automation tools and continuous monitoring solutions that provide visibility into the state of assets, user activities, and network behavior. Stakeholders will be able to use these tools to aggregate and analyze relevant security and operational data to understand the organizations security posture, quantify business risk, and make informed decisions that support organizational objectives while protecting critical information. Organizations will be able to use these tools to augment and automate information sharing activities to collaborate with partners to identify and mitigate threats. Other automation tools will be able to integrate with these capabilities to enforce policies based on human decisions to harden systems, prevent misuse and reduce the overall attack surface.

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. Key Concepts

The operational methods we use within the bounds of our present realities are failing us - we are falling behind. We have begun to recognize that the evolution of threat agents, increasing system complexity, rapid situational security change, and scarce resources are detrimental to our success. There have been efforts to remedy our circumstance, and these efforts are generally known as "Security Automation."

Security Automation is a general term used to reference standards and specifications originally created by the National Institute of Standards and Technology (NIST) and/or the MITRE Corporation. Security Automation generally includes languages, protocols (prescribed ways by which specification collections are used), enumerations, and metrics.

These specifications have provided an opportunity for tool vendors and enterprises building customized solutions to take the appropriate steps toward enabling Security Automation by defining common information expressions. In effect, common expression of information enables interoperability between tools (whether customized, commercial, or freely available). Another important capability common expression provides is the ability to automate portions of security processes to gain efficiency, react to new threats in a timely manner, and free up security personnel to work on more advanced problems within the processes in which they participate.

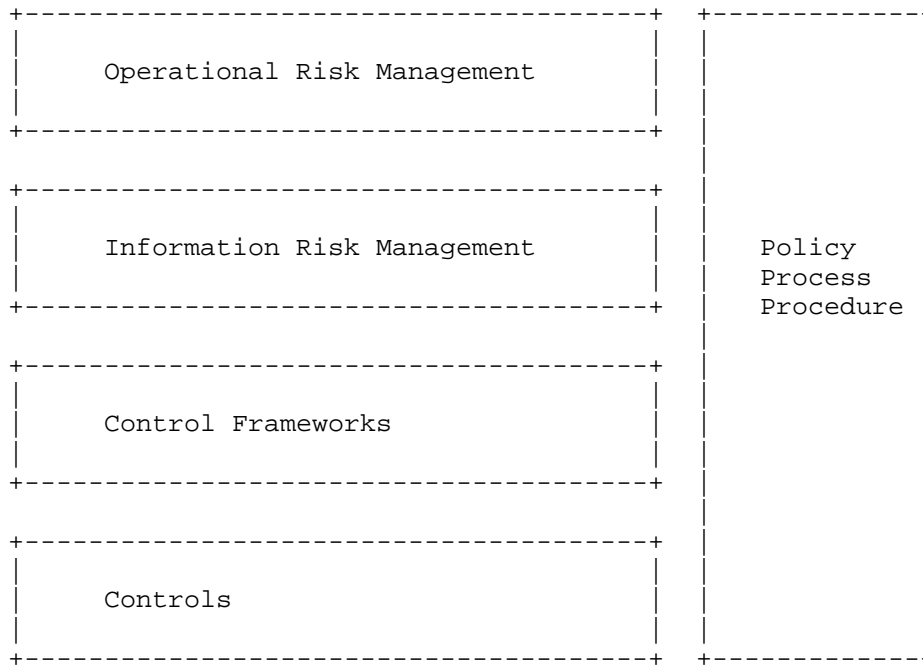


Figure 1

The figure above provides some context for our focus area. Organizations of all sizes will have a more or less formal risk management program, depending upon their maturity and organization-specific needs. A small business with only a few employees may not have a formally recognized risk management program, but they still lock the doors at night. Typically, financial entities and governments sit at the other end of the spectrum with often large, laborious risk frameworks. The point is that all organizations practice, to some degree, Operational Risk Management. An Information Risk Management program is most likely a constituent of Operational Risk Management (another constituent might be Financial Risk Management). In the Information Risk Management domain, we often use Control Frameworks to provide guidance for organizations practicing ORM in an information context, and these Control Frameworks define a variety of Controls.

From ORM, IRM, Control Frameworks, and the Controls themselves, organizations derive a set of organization-specific policies, processes, and procedures. Such policies, processes, and procedures make use of a library of supporting information commonly stipulated by the organization (i.e. enterprise acceptable use policies), but

often prescribed by external entities (i.e. Payment Card Industry Data Security Standards, Sarbanes-Oxley, or EU Data Privacy Directive). The focus of this document spans Controls, certain aspects of policy, process, and procedure, and Control Frameworks.

3. Use Cases

This document addresses three use cases: System State Assessment, Enforcement of Acceptable State, Security Control Verification and Monitoring.

3.1. UC1: System State Assessment

3.1.1. Goal

Assess security state of a given system to be in compliance with enterprise standards and, therefore, ensure alignment with enterprise policy.

3.1.2. Main Success Scenario

1. Define target system to be assessed
2. Select acceptable state policies to apply to defined target
3. Collect actual state values from target
4. Compare actual state values collected from target with expected state values as expressed in acceptable state policies

3.1.3. Extensions

None.

3.2. UC2: Enforcement of Acceptable State

3.2.1. Goal

Allow or deny access to a desired resource based on system characteristics compliance with enterprise policy.

3.2.2. Main Success Scenario

1. An entity (user on a system or the system itself) requests access to a given resource (i.e. network connection)

2. Assessment of system state is achieved using Section 3.1
3. Based on assessment results (i.e. compliance level with enterprise policy)
 - A. System is allowed access to requested resource, or
 - B. System is denied access to requested resource
- 3.2.3. Extensions

None.
- 3.3. UC3: Security Control Verification and Monitoring
 - 3.3.1. Goal

Continuous assessment of the implementation and effectiveness of security controls based on machine processable content.
 - 3.3.2. Main Success Scenario
 1. Define set of targets to be assessed.
 2. Select acceptable state policies to apply to set of targets
 3. Define assessment trigger based on either a
 - A. Time period, or
 - B. System/enterprise event.
 4. Define result reporting/alerting criteria
 5. Enable continuous assessment
 - 3.3.3. Extensions

None.
4. Functional Capabilities

In general, the activities of managing assets, configurations, and vulnerabilities are common between UC1 and UC2. UC1 uses these activities to either grant or deny an entity access to a requested resource. UC2 uses these activities in support of compliance measurement on a periodic basis.

At the most basic level, an enterprise needing to satisfy UC1 and UC2 will need certain capabilities to be met. Specifically, we are talking about risk management capabilities. This is the central problem domain, so it makes sense to be able to convey information about technical and non-technical controls, benchmarks, control requirements, control frameworks and other concepts in a common way.

4.1. Capabilities Supporting UC1

As described in Section Section 3.1, the required capabilities need to support assessing host and/or network state in an automated manner. This is, essentially, a configuration assessment check before allowing a full connection to the network.

4.1.1. Asset Management

Effective Asset Management is a critical foundation upon which all else in risk management is based. There are two important facets to asset management: 1) understanding coverage (how many assets are under control) and, 2) understanding specific asset details. Coverage is fairly straightforward - assessing 80% of the enterprise is better than assessing 50% of the enterprise. Getting asset details is comparatively subtle - if an enterprise does not have a precise understanding of its assets, then all acquired data and consequent actions are considered suspect. Assessing assets (managed and unmanaged) requires that we see and properly characterize our assets at the outset and over time.

What we need to do initially is discover and characterize our assets, and then identify them in a common way. Characterization may take the form of logical characterization or security characterization, where logical characterization may include business context not otherwise related to security, but which may be used as information in support of decision making later in risk management workflows.

The following list details the requisite Asset Management capabilities (later described in Section 5):

- o Discover assets in the enterprise
- o Characterize assets according to security and non-security asset properties
- o Identify and describe assets using a common vocabulary between implementations
- o Reconcile asset representations originating from disparate tools

- o Manage asset information throughout the asset's life cycle

4.1.2. Data Collection

Related to managing assets, and central to any automated assessment solution is the ability to collect data from target hosts (some might call this "harvesting"). Of particular interest are data representing the security state of a target, be it a computing device, network hardware, operating system, or application. The primary interest of the activities demanding data collection is centered on object state collection, where objects may be file attributes, operating system and/or application configuration items, and network device configuration items among others.

4.1.2.1. Security Configuration Management

There are many valid perspectives to take when considering required capabilities, but the industry seems to have roughly settled upon the notion of "Security Configuration Management" (there are variants of the term). Security Configuration Management (SCM) is a simple way to reference several supporting capabilities involving technical and non-technical assessment of systems.

The following capabilities support SCM:

- o Target Assessment

- * Collect the state of non-technical controls commonly called administrative controls (i.e. policy, process, procedure)
- * Collect the state of technical controls including, but not necessarily limited to:
 - + Target configuration items
 - + Target patch level
 - + Target object state

4.1.2.2. Vulnerability Management

SCM is only part of the solution, as it deals exclusively with the configuration of computing devices, including software vulnerabilities (by testing for patch levels). All vulnerabilities need to be addressed as part of a comprehensive risk management program, which is a superset of software vulnerabilities. Thus, the capability of assessing non-software vulnerabilities applicable to the in-scope system is required.

The following capabilities support Vulnerability Management:

1. Assessment

- * Non-technical Vulnerability Assessment (i.e. interrogative)
- * Technical Vulnerability Assessment

4.1.3. Assessment Result Analysis

At the most basic level, the data collected needs to be analyzed for compliance to a standard stipulated by the enterprise. Such standards vary between enterprises, but commonly take a similar form.

The following capabilities support the analysis of assessment results:

- o Comparing actual state to expected state
- o Scoring/weighting individual comparison results
- o Relating specific comparisons to benchmark-level requirements
- o Relating benchmark-level requirements to one or more control frameworks

4.1.4. Content Management

It should be clear by now that the capabilities required to support risk management state measurement will yield volumes of content. The efficacy of risk management state measurement depends directly on the stability of the driving content, and, subsequently, the ability to change content according to enterprise needs.

Capabilities supporting Content Management should provide the ability to create/define or modify content, as well as store and retrieve said content of at least the following types:

- o Configuration Standards
- o Scoring Models
- o Vulnerability Information
- o Patch Information
- o Asset Characterization

Note that the ability to modify content is in direct support of tailoring content for enterprise-specific needs.

4.2. Capabilities Supporting UC2

UC2 is dependent upon UC1 and, therefore, includes all of the capabilities described in Section Section 4.1. UC2 describes the ability to make a resource access decision based on an assessment of the requesting system (either by the system itself or on behalf of a user operating that system). There are two chief capabilities required to meet the needs expressed in Section Section 3.2: Assessment Query and Transport, and Acceptable State Enforcement.

4.2.1. Assessment Query and Transport

Under certain circumstances, the system requesting access may be unknown, which can make querying the system problematic (consider a case where a system is connecting to the network and has no assessment software installed). Note that The Network Endpoint Assessment (NEA) protocols (PA-TNC [RFC5792], PB-TNC [RFC5793], PT-TLS [I-D.ietf-nea-pt-tls], and PT-EAP [I-D.ietf-nea-pt-eap]) may be used to query and transport the things to be measured.

4.2.2. Acceptable State Enforcement

Once the assessment has been performed a decision to allow or deny access to the requested resource can be made. Making this decision is a necessary but insufficient condition for enforcement of acceptable state, and an implementation must have the ability to actively allow or deny access to the requested resource. For example, network enforcement may be implemented with RADIUS [RFC2865] or DIAMETER [RFC3588].

4.3. Capabilities Supporting UC3

Recall that UC3 is dependent upon UC1 and therefore includes all of the capabilities described in Section 4.1. The difference in UC3 is the notion of when to assess rather than what to assess. Therefore, the capabilities described in this section are relevant only to the "when" and not to the "what."

4.3.1. Tasking and Scheduling

The ability to task and schedule assessments is requisite for any effective risk management program. Tasking refers to the ability to create a set of instructions to be conveyed at a later time via scheduling. Tasking, therefore, involves selecting a set of assessment criteria, assigning that set to a group of assets, and

expressing that information in a manner that can be consumed by a collection tool. Scheduling comes into play when the enterprise determines when to perform a specific assessment task (or set of tasks). Scheduling may be expressed in a way that constrains tasks to execute only during defined periods, can be ad hoc, or may be triggered by the analysis of previous assessment results or events detected in the enterprise.

The following capabilities support Tasking and Scheduling:

- o Selection of assessment criteria
- o Defining in-scope assets (i.e. targeting)
- o Defining periodic assessments for a given set of tasks
- o Defining assessment triggers for a given set of tasks

4.3.2. Data Aggregation and Reporting

Assessment results are produced for every asset assessed, and these results must be reported not only individually, but in the aggregate, and in accordance with enterprise needs. Enterprises should be able to aggregate and report on the data their assessments produce in a number of different ways in order to support different levels of decision making. At times, security operations personnel may be interested in understanding where the most critical risks exist in their enterprise so as to focus their remediation efforts in the most effective way (in terms of cost and return). At other times, only aggregated scores will matter, as might be the case when reporting to an information security manager or other executive-level role.

It is not the position of these capabilities to provide explicit details about how reports should be formatted for presentation, but only what information they should contain for a particular purpose. Furthermore, it is quite easy to imagine the need for a capability providing extensibility to aggregation and reporting.

Aggregating assessment results by the following capabilities supports Data Aggregation and Reporting

- o By asset characterization
- o By assessment criteria
- o By control framework

- o By benchmark
- o By other attributes/properties of assessment characteristics
- o Extensible aggregation and reporting

5. Functional Components

This section describes the functional components alluded to in the previous section Section 4. In keeping with the organization of the previous section, the following high-level functional capabilities are decomposed herein: Asset Management, Security Configuration Management, Vulnerability Management, Content Management, Assessment Result Analysis, Tasking and Scheduling, and Data Aggregation and Reporting.

5.1. Asset Management

As previously mentioned, asset management is a critically important component of any risk management program. If you stop to consider the different tools used to support a risk management program (i.e. IDS/IPS, Firewalls, NAC devices, WAFs, SCM, and so on), they all need, to some degree, an element of asset management. In this context, asset management is defined as the maintenance of necessary and accurate asset characteristics. Management of assets requires the ability to discover, characterize, and subsequently identify assets across enterprise tools. The components described herein support Section 4.1.1

5.1.1. Discovery

5.1.2. Characterization

5.1.2.1. Logical

5.1.2.2. Security

5.1.3. Asset Identification

5.2. Security Configuration Management

The components described herein support Section 4.1.2

5.2.1. Configuration Assessment

5.2.1.1. Non-technical Assessment

5.2.1.2. Technical Assessment

5.2.1.2.1. Configuration Assessment

5.2.1.2.2. Patch Assessment

5.2.1.2.3. Object State Assessment

5.3. Vulnerability Management

The components described herein support Section 4.1.2

5.3.1. Non-technical Vulnerability Assessment

5.3.2. Technical Vulnerability Assessment

5.4. Content Management

The components described herein support Section 4.1.4

5.4.1. Control Frameworks

5.4.2. Configuration Standards

5.4.3. Scoring Models

5.4.4. Vulnerability Information

5.4.5. Patch Information

5.4.6. Asset Information

5.5. Assessment Result Analysis

The components described herein support Section 4.1.3

5.5.1. Comparing Actual to Expected State

5.5.2. Scoring Comparison Results

5.5.3. Relating Comparison Results to Requirements

5.5.4. Relating Requirements to Control Frameworks

5.6. Tasking and Scheduling

The components described herein support Section 4.3.1

5.6.1. Selection of Assessment Criteria

5.6.2. Defining In-scope Assets

5.6.3. Defining Periodic Assessments

5.6.4. Defining Assessment Triggers

5.7. Data Aggregation and Reporting

The components described herein support Section 4.3.2

5.7.1. By Asset Characterization

5.7.2. By Assessment Criteria

5.7.3. By Control Framework

5.7.4. By Benchmark

5.7.5. By Ad Hoc/Extended Properties

6. Data Exchange Models and Communications Protocols

Document where existing work exists, what is currently defined by SDOs, and any gaps that should be addressed. Point to existing event, incident and network management standards when available. Describe emerging efforts that may be used for the creation of new standards. For gaps provide insight into what would be a good fit for SACM or another IETF working groups.

This will help us to identify what is needed for SACM to be successful. This section will help determine which of the specifications can be normatively referenced and what needs to be addressed in the IETF. This should help us determine any protocol or guidance documentation we will need to generate to support the described use cases.

Things to address:

For IETF related efforts, discuss work in NEA and MILE working groups. Address SNMP, NetConf and other efforts as needed.

Reference any Security Automation work that is applicable.

6.1. Data Exchange Models

The functional capabilities described in Section 4 require a significant number of models to be selected or defined in order to meet the needs of the three use cases presented in Section 3. A "model" in this sense is a logical arrangement of information that may have more than one syntactic binding. For the purpose of this document, only the logical data model is considered. However, where appropriate, example data models that may have well-defined syntactic expressions may be referenced.

6.1.1. Control Expression

For each we need an identification method, a logical expression and one or more syntactic bindings to that expression. For some, we may wish to associate a method of risk scoring.

6.1.1.1. Technical Control Expression

6.1.1.2. Non-technical Control Expression

6.1.1.2.1. Configuration Controls

6.1.1.2.2. Patches

6.1.1.2.3. Vulnerabilities

6.1.1.2.4. Object (Non-security) State

6.1.2. Control Frameworks

6.1.2.1. Logical Expression and Syntactic Binding(s)

6.1.2.2. Relationships

6.1.2.3. Substantiation (Control Requirement)

6.1.2.4. Reporting

6.1.3. Asset Expressions

6.1.3.1. Asset Identification

6.1.3.2. Asset Classification (Type)

- 6.1.3.3. Asset Attributes
 - 6.1.3.3.1. Criticality
 - 6.1.3.3.2. Classification (security)
 - 6.1.3.3.3. Owner
- 6.1.3.4. Information Expression (non-identifying)
- 6.1.3.5. Reporting
- 6.1.4. Benchmark/Checklist Expression
 - 6.1.4.1. Logical Expression and Bindings
 - 6.1.4.2. Checking Systems
 - 6.1.4.3. Results and Scoring
 - 6.1.4.4. Reporting
- 6.1.5. Check Language
 - 6.1.5.1. Logical Expression and Syntactic Binding(s)
 - 6.1.5.1.1. Technical
 - 6.1.5.1.2. Non-technical
 - 6.1.5.2. Reporting
- 6.1.6. Targeting Expression
 - 6.1.6.1. Information Owner
 - 6.1.6.2. System Owner
 - 6.1.6.2.1. Computing Device(s)
 - 6.1.6.2.2. Network(s)
 - 6.1.6.3. Assessor
 - 6.1.6.4. Computing Device
 - 6.1.6.5. Targeting Extensibility

6.2. Communication Protocols

6.2.1. Asset Management Interface

7. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see RFC 5226 [RFC5226] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

8. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

9. Acknowledgements

The author would like to thank Kathleen Moriarty and Stephen Hanna for contributing text to this document. The author would also like to acknowledge the members of the SACM mailing list for thier keen and insightful feedback on the concepts and text within this document.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

[I-D.ietf-nea-pt-eap]
Cam-Winget, N. and P. Sangster, "PT-EAP: Posture Transport (PT) Protocol For EAP Tunnel Methods",
draft-ietf-nea-pt-eap-02 (work in progress), May 2012.

[I-D.ietf-nea-pt-tls]
Sangster, P., Cam-Winget, N., and J. Salowey, "PT-TLS: A

TCP-based Posture Transport (PT) Protocol",
draft-ietf-nea-pt-tls-05 (work in progress), May 2012.

- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson,
"Remote Authentication Dial In User Service (RADIUS)",
RFC 2865, June 2000.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC
Text on Security Considerations", BCP 72, RFC 3552,
July 2003.
- [RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J.
Arkko, "Diameter Base Protocol", RFC 3588, September 2003.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs", BCP 26, RFC 5226,
May 2008.
- [RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute
(PA) Protocol Compatible with Trusted Network Connect
(TNC)", RFC 5792, March 2010.
- [RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC:
A Posture Broker (PB) Protocol Compatible with Trusted
Network Connect (TNC)", RFC 5793, March 2010.

Appendix A. Additional Stuff

This becomes an Appendix if needed.

Authors' Addresses

David Waltermire (editor)
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland 20877
USA

Phone:

Email: david.waltermire@nist.gov

Adam W. Montville
Tripwire, Inc.
101 SW Main Street, Suite 1500
Portland, Oregon 97204
USA

Phone:
Email: amontville@tripwire.com

