

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 4, 2015

D. Waltermire, Ed.
NIST
K. Watson
DHS
July 3, 2014

Information Model for Endpoint Assessment
draft-wandw-sacm-information-model-00

Abstract

This document proposes a draft information model for endpoint posture assessment. It describes the information needed to perform certain assessment activities.

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 January 4, 2015.

Copyright Notice

Copyright (c) 2014 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
2.	Problem Statement	5
2.1.	Problem Scope	5
2.2.	Mapping to SACM Use Cases	6
3.	Conventions used in this document	7
3.1.	Requirements Language	7
4.	Terms and Definitions	7
4.1.	Pre-defined and Modified Terms	7
4.2.	New Terms	8
5.	Foundational Concepts	8
5.1.	Core Principles	8
5.2.	Architecture Assumptions	9
6.	Endpoint Management	13
6.1.	Core Information Need	13
6.2.	Process Area Description	13
6.3.	Endpoint Management Process Operations	14
6.4.	Information Model Requirements	14
6.4.1.	Enroll Operation	14
7.	Software Management	15
7.1.	Core Information Needs	15
7.2.	Process Area Description	16
7.3.	Software Management Process Operations	18
7.4.	Information Model Requirements	18
7.4.1.	Define Guidance	19
7.4.2.	Collect Inventory	19
7.4.3.	Evaluate Software Inventory Posture	20
7.4.4.	Report Evaluation Results	21
8.	Configuration Management	21
8.1.	Core Information Needs	22
8.2.	Process Area Description	22
8.2.1.	The Existence of Configuration Item Guidance	23
8.2.2.	Configuration Collection Guidance	24
8.2.3.	Configuration Evaluation Guidance	24
8.2.4.	Local Configuration Management Process	24
8.3.	Configuration Management Operations	26
8.4.	Information Model Requirements	26
8.4.1.	Define Guidance	26
8.4.2.	Collect Posture Attributes Operation	28
8.4.3.	Evaluate Posture Attributes Operation	29
8.4.4.	Report Evaluation Results Operation	30
9.	Vulnerability Management	31

9.1.	Core Information Needs	31
9.2.	Process Area Description	31
9.3.	Vulnerability Management Process Operations	32
9.4.	Information Model Requirements	32
9.4.1.	Collect Vulnerability Reports	32

9.4.2.	Evaluate Vulnerability Posture	33
9.4.3.	Report Evaluation Results	35
10.	From Information Needs to Information Elements	35
11.	Information Model Elements	36
11.1.	Asset Identifiers	37
11.1.1.	Endpoint Identification	40
11.1.2.	Software Identification	41
11.1.3.	Hardware Identification	44
11.1.4.	Hardware Identification	44
11.2.	Other Identifiers	44
11.2.1.	Platform Configuration Item Identifier	44
11.2.2.	Configuration Item Identifier	50
11.2.3.	Vulnerability Identifier	52
11.3.	Endpoint characterization	52
11.4.	Posture Attribute Expression	56
11.4.1.	Platform Configuration Attributes	56
11.4.2.	Platform Configuration Attributes	56
11.5.	Actual Value Representation	58
11.5.1.	Software Inventory	58
11.5.2.	Collected Platform Configuration Posture Attributes	59
11.6.	Evaluation Guidance	60
11.6.1.	Configuration Evaluation Guidance	60
11.7.	Evaluation Result Reporting	62
11.7.1.	Configuration Evaluation Results	62
11.7.2.	Software Inventory Evaluation Results	64
12.	Acknowledgements	64
13.	IANA Considerations	65
14.	Security Considerations	65
15.	References	65
15.1.	Normative References	65
15.2.	Informative References	65
15.3.	URIs	69
	Authors' Addresses	69

[1.](#) Introduction

The posture of an endpoint is the status of an endpoint's assets with respect to the security policies and risk models of the organization.

A system administrator needs to be able to determine which of the collection of assets that constitute an endpoint have a security problem and which do not conform the organization's security policies. The CIO needs to be able to determine whether endpoints have security postures that conform to the organization's policies to ensure that the organization is complying with its fiduciary and regulatory responsibilities. The regulator or auditor needs to be able to assess the level of due diligence being achieved by an organization to ensure that all regulations and due diligence expectations are being met. The operator needs to understand which

assets have deviated from organizational policies so that those assets can be remedied.

Operators will focus on which endpoints are composed of specific assets with problems. CIO and auditors need a characterization of how an organization is performing as a whole to manage the posture of its endpoints. All of these actors need deployed capabilities that implement security automation standards in the form of data formats, interfaces, and protocols to be able to assess, in a timely and secure fashion, all assets on all endpoints within their enterprise. This information model provides a basis to identify the desirable characteristics of data models to support these scenarios. Other SACM specifications, such as the SACM Architecture, will describe the potential components of an interoperable system solution based on the SACM information model to address the requirements for scalability, timeliness, and security.

This draft was developed in response to the Call for Contributions for the SACM Information Model sent to NIST [[IM-LIAISON-STATEMENT-NIST](#)]. This draft proposes a notional information model for endpoint posture assessment. It describes the information needed to perform certain assessment activities and relevant work that may be used as a basis for the development of specific data models. The terms information model and data model loosely align with the terms defined in [RFC3444](#) [[RFC3444](#)].

The four primary activities to support this information model are:

1. Endpoint Identification

2. Endpoint Characterization
3. Endpoint Attribute Expression/Representation
4. Policy evaluation expression and results reporting

These activities are aimed at the level of the technology that performs operations to support collection, evaluation, and reporting.

Review of the SACM Use Case [[I-D.ietf-sacm-use-cases](#)] usage scenarios show a common set of business process areas that are critical to understanding endpoint posture such that appropriate policies, security capabilities, and decisions can be developed and implemented.

For this information model we have chosen to focus on the following business process areas:

Waltermire & Watson	Expires January 4, 2015	[Page 4]
---------------------	-------------------------	----------

Internet-Draft	Endpoint Assessment Information Model	July 2014
----------------	---------------------------------------	-----------

- o Endpoint Management
- o Software Management
- o Configuration Management
- o Vulnerability Management

These management process areas are a way to connect the SACM use cases and building blocks [[I-D.ietf-sacm-use-cases](#)] to the organizational needs such that the definition of information requirements has a clearly understood context.

2. Problem Statement

Scalable and sustainable collection, expression, and evaluation of endpoint information is foundational to SACM's objectives. To secure and defend one's network one must reliably determine what devices are on the network, how those devices are configured from a hardware perspective, what software products are installed on those devices, and how those products are configured. We need to be able to determine, share, and use this information in a secure, timely,

consistent, and automated manner to perform endpoint posture assessments.

This represents a large and broad set of mission and business processes, and to make the most effective of use of technology, the same data must support multiple processes. The activities and processes described within this memo tend to build off of each other to enable more complex characterization and assessment. In an effort to create an information model that serves a common set of management processes represented by the usage scenarios in the SACM Use Cases document, we have narrowed down the scope of this model.

[2.1.](#) Problem Scope

The goal of this first iteration of the information model is to define the information needs for an organization to effectively manage the endpoints operating on their network, the software installed on those endpoints, and the configuration of that software. Once we have those three business processes in place, we can then identify vulnerable endpoints in a very efficient manner.

The four business process areas represent a large set of tasks that support endpoint posture assessment. In an effort to address the most basic and foundational needs, we have also narrowed down the scope inside of each of the business processes to a set of defined

tasks that strive to achieve specific results in the operational environment and the organization. These tasks are:

1. Define the assets. This is what we want to know about an asset. For instance, organizations will want to know what software is installed and its many critical security attributes such as patch level.
2. Resolve what assets actually compose an endpoint. This requires populating the data elements and attributes needed to exchange information pertaining to the assets composing an endpoint.
3. Express what expected values for the data elements and attributes need to be evaluated against the actual collected instances of asset data. This is how an organization can express its policy

for an acceptable data element or attribute value. A system administrator can also identify specific data elements and attributes that represent problems, such as vulnerabilities, that need to be detected on an endpoint.

4. Evaluate the collected instances of the asset data against those expressed in the policy.
5. Report the results of the evaluation.

[2.2](#). Mapping to SACM Use Cases

This information model directly corresponds to all four use cases defined in the SACM Use Cases draft [[I-D.ietf-sacm-use-cases](#)]. It uses these use cases in coordination to achieve a small set of well-defined tasks.

Sections [6](#) thru 9 address each of the process areas. For each process area, a "Process Area Description" sub-section represent an end state that is consistent with all the General Requirements and many of the Use Case Requirements identified in the requirements draft [[I-D.camwinget-sacm-requirements](#)].

The management process areas and supporting operations defined in this memo directly support REQ004 Endpoint Discovery; REQ005-006 Attribute and Information Based Queries, and REQ0007 Asynchronous Publication.

In addition, the operations that defined for each business process in this memo directly correlate with the typical workflow identified in the SACM Use Case document.

[3](#). Conventions used in this document

[3.1](#). 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](#)].

[4.](#) Terms and Definitions

This section describes terms that have been defined by other RFCs and Internet Drafts, as well as new terms introduced in this document.

[4.1.](#) Pre-defined and Modified Terms

This section contains pre-defined terms that are sourced from other IETF RFCs and Internet Drafts. Descriptions of terms in this section will reference the original source of the term and will provide additional specific context for the use of each term in SACM. For sake of brevity, terms from [[I-D.ietf-sacm-terminology](#)] are not repeated here unless the original meaning has been changed in this document.

Asset For this Information Model it is necessary to change the scope of the definition of asset from the one provided in [[I-D.ietf-sacm-terminology](#)]. Originally defined in [[RFC4949](#)] and referenced in [[I-D.ietf-sacm-terminology](#)] as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission." This definition generally relates to an "IT Asset", which in the context of this document is overly limiting. For use in this document, a broader definition of the term is needed to represent non-IT asset types as well.

In [[NISTIR-7693](#)] an asset is defined as "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)." This definition aligns better with common dictionary definitions of the term and better fits the needs of this document.

[4.2.](#) New Terms

IT Asset Originally defined in [\[RFC4949\]](#) as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission."

Security Content Automation Protocol (SCAP) According to SP800-126, SCAP, pronounced "ess-cap", is "a suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans." SP800-117 revision 1 [\[SP800-117\]](#) provides a general overview of SCAP 1.2. The 11 specifications that comprise SCAP 1.2 are synthesized by a master specification, SP800-126 revision 2 [\[SP800-126\]](#), that addresses integration of the specifications into a coherent whole. The use of "protocol" in its name is a misnomer, as SCAP defines only data models. SCAP has been adopted by a number of operating system and security tool vendors.

[5.](#) Foundational Concepts

[5.1.](#) Core Principles

This information model is built on the following core principles:

- o Collection and Evaluation are separate tasks.
- o Collection and Evaluation can be performed on the endpoint, at a local server that communicates directly with the endpoint, or based on data queried from a back end data store that does not communicate directly with any endpoints.
- o Every entity (human or machine) that notifies, queries, or responds to any guidance, collection, or evaluation task must have a way of identifying itself and/or presenting credentials. Authentication is a key step in all of the processes, and while needed to support the business processes, information needs to support authentication are not highlighted in this information model. There is already a large amount of existing work that defines information needs for authentication.
- o Policies are reflected in guidance for collection, evaluation, and reporting.
- o Guidance will often be generated by humans or through the use of transformations on existing automation data. In some cases,

guidance will be generated dynamically based on shared information or current operational needs. As guidance is created it will be published to an appropriate guidance data store allowing guidance to be managed in and retrieved from convenient locations.

- o Operators of a continuous monitoring or security automation system will need to make decisions when defining policies about what guidance to use or reference. The guidance used may be directly associated with policy or may be queried dynamically based on associated metadata.
- o Guidance can be gathered from multiple data stores. It may be retrieved at the point of use or may be packaged and forwarded for later use. Guidance may be retrieved in event of a collection or evaluation trigger or it may be gathered ahead of time and stored locally for use/reference during collection and evaluation activities.

[5.2.](#) Architecture Assumptions

This information model will focus on WHAT information needs to be exchanged to support the business process areas. The architecture document is the best place to represent the HOW and the WHERE this information is used. In an effort to ensure that the data models derived from this information model scale to the architecture, four core architectural components need to be defined. They are producers, consumers, capabilities, and repositories. These elements are defined as follows:

- o Producers (e.g., Evaluation Producer) collect, aggregate, and/or derive information items and provide them to consumers. For this model there are Collection, Evaluation, and Results Producers. There may or may not be Guidance Producers.
- o Consumers (e.g., Collection Consumer) request and/or receive information items from producers for their own use. For this model there are Collection, Evaluation, and Results Consumers. There may or may not be Guidance Consumers.
- o Capabilities (e.g., Posture Evaluation Capability) take the input from one or more producers and perform some function on or with that information. For this model there are Collection Guidance, Collection, Evaluation Guidance, Evaluation, Reporting Guidance, and Results Reporting Capabilities.
- o Repositories (e.g., Enterprise Repository) store information items

that are input to or output from Capabilities, Producers, and

Consumers. For this model we refer to generic Enterprise and Guidance Repositories.

Information that needs to be communicated by or made available to any of these components will be specified in each of the business process areas.

In the most trivial example, illustrated in Figure 1, Consumers either request information from, or are notified by, Producers.

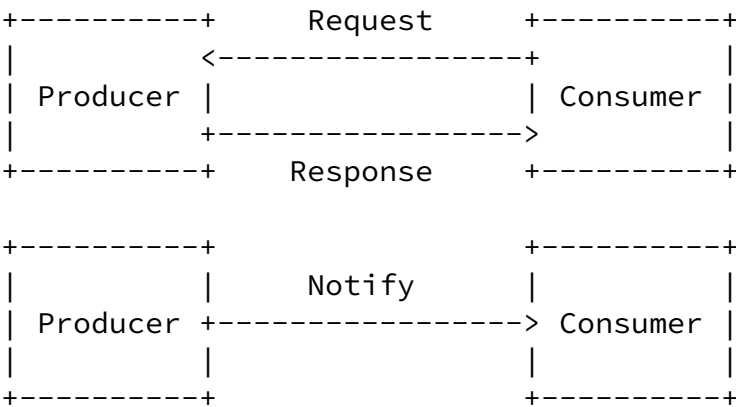


Figure 1: Example Producer/Consumer Interactions

As illustrated in Figure 2, writing and querying from data repositories are a way in which this interaction can occur in an asynchronous fashion.

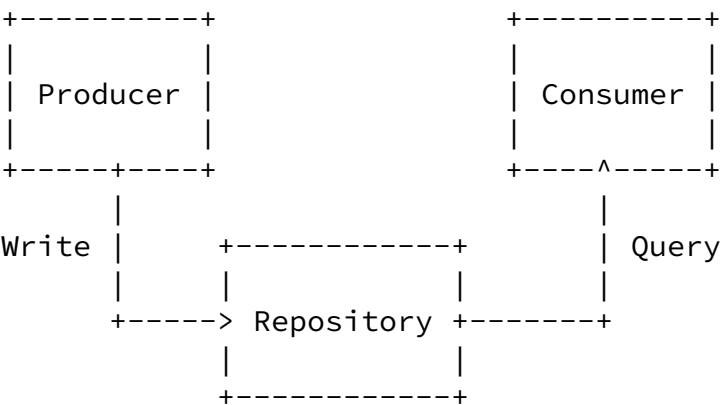


Figure 2: Producer/Consumer Repository Interaction

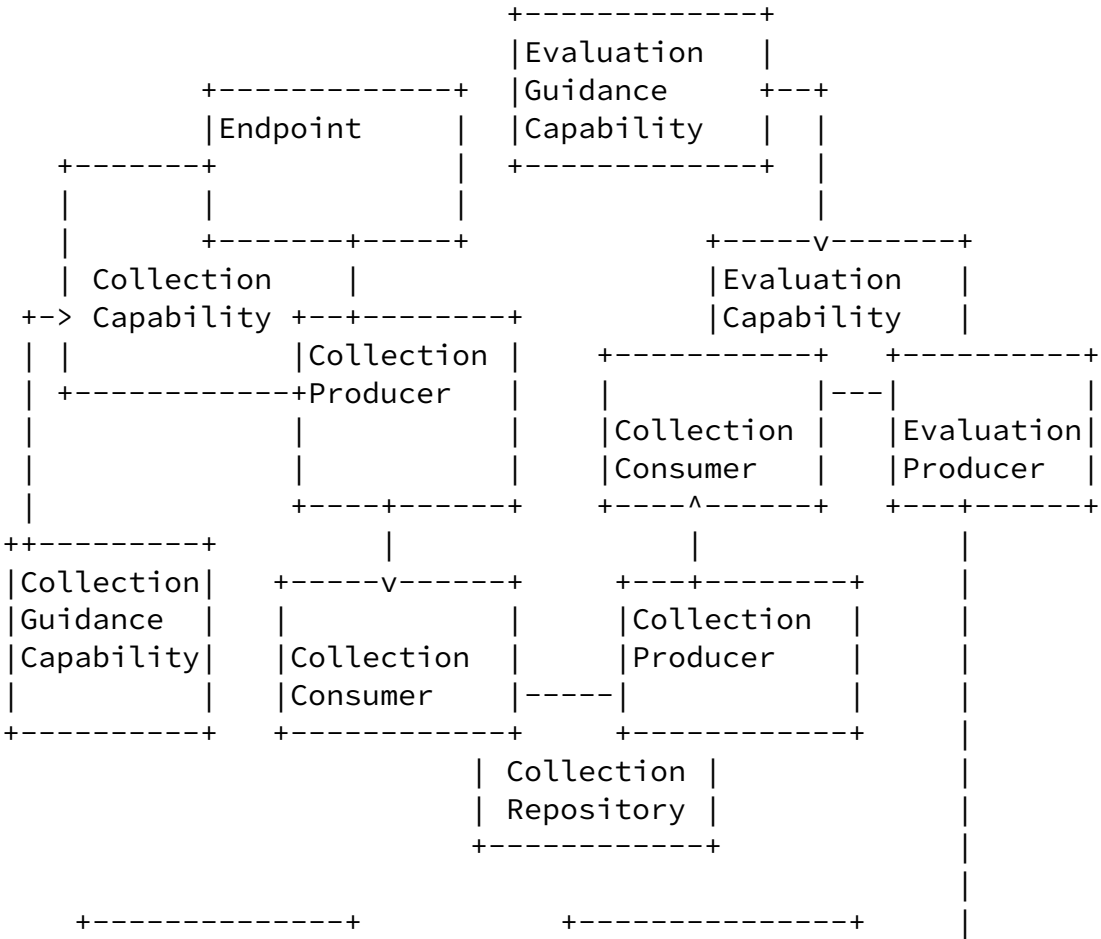
To perform an assessment, these elements are chained together. The diagram below is illustrative of this and process, and is meant to demonstrate WHAT basic information exchanges need to occur, while trying to maintain flexibility in HOW and WHERE they occur.

For example:

Waltermire & Watson	Expires January 4, 2015	[Page 10]
---------------------	-------------------------	-----------

Internet-Draft	Endpoint Assessment Information Model	July 2014
----------------	---------------------------------------	-----------

- o the collection capability can reside on the endpoint or not.
- o the collection producer can be part of the collection capability or not.
- o a repository can be directly associated with a producer and/or an evaluator or stand on its own.
- o there can be multiple "levels" of producers and consumers.



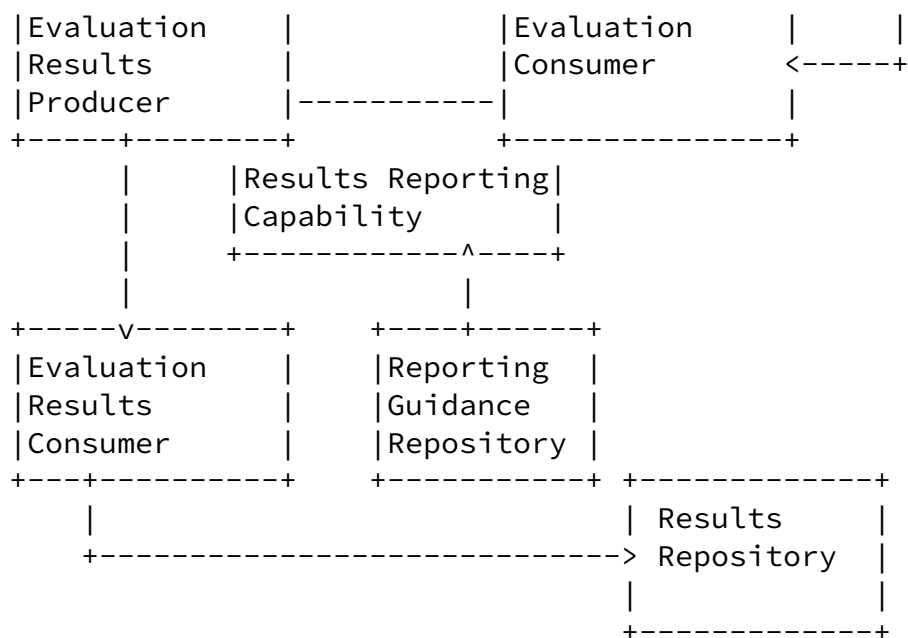


Figure 3: Producer/Consumer Complex Example

This illustrative example in Figure 3 provides a set of information exchanges that need to occur to perform a posture assessment. The rest of this information model is using this set of exchanges based

on these core architectural components as the basis for determining information elements.

6. Endpoint Management

6.1. Core Information Need

Unique Endpoint Identifier: The organization needs to uniquely identify and label an endpoint, regardless of if it is known about a priori or discovered it in the operational environment.

6.2. Process Area Description

The authors envisage a common "lifecycle" for all endpoints in an enterprise network. Each endpoint's lifecycle begins with an "enrollment" operation, where a description of the endpoint is added to the enterprise repository of "known endpoints." The enrollment

operation may be performed manually, in advance of an endpoint's first connection request, or automatically at the time of an endpoint's first connection request.

Manual enrollment is typically done in situations where endpoint devices are issued by the enterprise (as contrasted with "bring your own device" situations), and must first be configured by the enterprise's Information Technology (IT) department before they are allowed to connect to the network. When enrollment is performed in this manner, administrators typically know a lot about the physical endpoint, such as any persistent identifying characteristics (e.g., its primary MAC address), its assigned IP address and planned physical location within the network, its role within the network (e.g., end-user workstation, database server, webserver, etc.), and the responsible parties (e.g., asset owner, device maintainer). These data elements may be associated with the endpoint when the endpoint characteristics are entered into an enterprise repository as part of the enrollment process.

For networks with fewer access restrictions (e.g., guest wireless networks, and "bring your own device" networks), enrollment may occur automatically when an endpoint device first attempts to connect to the network. In these situations, enrollment typically happens at machine speed, without a human administrator in the loop. As a result, much less may be known about the endpoint that is being enrolled, and thus only minimal data elements may be available for automatic entry into an enterprise repository.

[6.3.](#) Endpoint Management Process Operations

Waltermire et al. define an endpoint as "any physical or virtual device that may have a network address" (Waltermire, et al., 2014). Endpoint management encompasses all security automation processes involved in the tracking and monitoring of endpoints as devices which are connected (even if only transiently) to an enterprise network. Based on the vision, we have identified the following operations for Endpoint Management:

1. Enroll/Expel: Add an endpoint to (or remove an endpoint from) the

list of known endpoints that may be allowed to access network resources.

[6.4.](#) Information Model Requirements

In this section we describe the data that enterprises will need to carry out for each endpoint management operation.

[6.4.1.](#) Enroll Operation

We allow for two modes of the "Enroll" operation: (1) manual, and (2) automatic. When the "enroll" operation is performed in the "manual" mode, enrollment occurs before the enrolled endpoint first attempts to connect to the enterprise network. The result of manual enrollment is that the enterprise repository is updated with records to indicate that the endpoint is enrolled and thus is "known". During enrollment, credentials may be issued (e.g., host or user certificates) and placed on the endpoint, to be furnished each time the endpoint attempts to connect to the network. We assume that as a byproduct of manual enrollment, the enterprise repository will contain a persistent unique identifier for the enrolled endpoint.

In the automatic mode, enrollment occurs as a side effect of a connection request. Here, the endpoint requesting access has not previously been manually enrolled and is thus "unknown". If policy permits, unknown endpoints may still be allowed to connect to the network and may be given limited access to resources.

To complete the enrollment operation, the following information elements may be needed ('M' indicates 'mandatory' and 'O' indicates 'optional'):

- o (M) Unique Endpoint Identifier: a persistent unique identifier for the endpoint
- o (O) Device Role: the organization needs to identify the intended use of the device (e.g., workstation, server, router).

- o (O) Asset Ownership: the organization needs to know what person and/or organization is responsible for maintaining the device.
- o (O) Other Identifying Characteristics: the organization must know

what other identifiers can be mapped to the Unique Endpoint Identifier (e.g., IP address, MAC address).

While many of these elements may be automatically collected, data pertaining to device role and ownership often require manual entry.

[7.](#) Software Management

This section presents an information model for managing information about software installed on endpoints. Software management encompasses the subset of tasks within security automation and continuous monitoring which play a role in compiling inventories of software products installed on endpoints and transmitting those inventories to software inventory data consumers. Software inventory data consumers may store the software inventory data and/or perform enterprise-level software inventory-related security posture assessments. While software enforcement policies that are invoked and enforced at the time of installation or execution are out of the scope of SACM, they require the same software guidance information to be produced and exchanged. For that reason, the first operation is to Define Guidance making it available to perform a posture evaluation activity during any operation.

[7.1.](#) Core Information Needs

Unique Endpoint Identifier: Organizations need to be able to relate the instances of software to the endpoint on which it is installed. This should be consistent with the identification of the endpoint when it is enrolled (see [Section 6.4.1](#)).

Unique Software Identifier: Organizations need to be able to uniquely identify and label software installed on an endpoint. Specifically, they need to know the name, publisher, unique ID, and version; and any related patches. In some cases the software's identity might be known a priori by the organization; in other cases, a software identity might be first detected by an organization when the software is first inventoried in an operational environment. Due to this, it is important that an organization have a stable and consistent means to identify software found during collection.

[7.2.](#) Process Area Description

The authors envisage an automated capability that will collect software inventory data on endpoints and transmit that data to interested software inventory data consumers. "Inventory data" is information about software products that may be operating systems, end-user applications, or other software-based systems and services—that have been added to, removed from, or modified on endpoints. The collection and transmission of inventory data from a given endpoint to an inventory data consumer may be scheduled, event-driven, or on demand based on the requested collection policy. That is, endpoints may be configured to collect and transmit inventory data on a predefined schedule (scheduled), in response to a change in inventory state (event-driven), or whenever an inventory consumer requests (on demand).

On any occasion when inventory data is collected and transmitted, the data may be "complete" or be an "update" to previously-transmitted data based on the requested collection policy. Inventory data is considered to be "complete" when it is intended to reflect a comprehensive and up-to-date enumeration of all software products that are believed to be installed on a given endpoint as of the time the inventory is taken. Inventory data is considered to be an "update" when the data is limited to documenting only the changes that have occurred since the last complete inventory or update. Ad hoc inventory data requests should also be supported; that is, an inventory data consumer should be able to issue ad hoc queries to an endpoint regarding specific identified products. Endpoints should be able to indicate whether or not an identified product is installed, and should be able to answer various questions about an installed software product including: the date/time it was most recently installed, removed or patched/updated, which patches are installed, and the names and properties (e.g., versions, hashes) of files associated with the product. Queries concerning the configuration of installed software products are addressed separately in [Section 7](#).

The author's vision rests on a model of the basic processes involved in software product installation and maintenance. We use the term "inventory event" to refer generically to any of three possible events involving software which may occur on endpoints: (1) installation (adding a software product to an endpoint's inventory), (2) modification (changing any files associated with a previously-installed product, and (3) removal (eliminating a software product from an endpoint's inventory).

Under this model, each endpoint may support (but is not required to) a resident "inventory manager". If present, the inventory manager is

an installed software product which provides a standard interface to

"product installers", which are specialized software applications that are designed to install, modify or remove other software products on endpoints. Product installers are generally expected to interact with the resident inventory manager, if one is present, but are not required to. By interacting with the inventory manager, product installers notify the inventory manager of any inventory events they are generating, and supply data values needed to characterize the event. When a product installer interacts correctly with a resident inventory manager, we say that it has generated a "conforming inventory event", meaning it has installed, modified or removed a software product in a manner that conforms to the inventory manager's expectations, as defined by its interface. When product installers fail to interact properly with a resident inventory manager, bypass it altogether, or when an inventory manager is not resident, we say that the resulting inventory event is "non-conforming". Additionally in the non-conforming case, a resident inventory manager may also monitor a filesystem or other installation contexts to detect changes to software to characterize the nature of the change.

On Linux systems, RPM and DPKG are examples of inventory managers. Each provides a standard product installer application which parses specially-formatted package files, updates the RPM/APT database, and copies files to their intended locations.

We require that when a product installer generates a conforming inventory event, the resident inventory manager shall update a local inventory data store on the endpoint. The local inventory data store must maintain an up-to-date record of all software products installed on the endpoint. The local inventory data store should also maintain a record of all inventory events, including product modifications and removals for later collection. The resident inventory manager should have the ability to provide event-driven notification to other software systems, to support reporting of inventory change events as soon as they occur.

Because our model allows for non-conforming inventory events, as well as for situations in which an endpoint does not support a resident inventory manager, we allow for some number of "endpoint scanners" to access endpoints either directly (by being resident on the endpoint

and by having the privileges necessary to inspect all areas of the endpoint where installed software may be present) or indirectly (e.g., by monitoring network traffic between the endpoint and other devices), and attempt to infer inventory events which may have happened at some point in the past.

Compiling a complete and accurate inventory of software products installed on an endpoint thus involves collecting information about

conforming as well as non-conforming inventory events. Information about conforming inventory events is obtained from the resident inventory manager, if present. Information about non-conforming inventory events is obtained by running one or more endpoint scanners. These tasks are performed by an "inventory producer", which may or may not reside on an endpoint, but which is able to interact with any resident inventory manager, as well as to initiate scans using available endpoint scanners. Inventory producers transmit collected inventory data to one or more inventory consumers, which may store that data in repositories for later assessment, perform assessments on that data directly, or both.

The collection and transmission of software inventory data is needed to enable assessment of security posture attributes associated with software inventory. For example, an enterprise may need to assess compliance with "whitelists" (lists of software products authorized for use on network devices) and "blacklists" (lists of specifically prohibited products). For another example, an enterprise may need to assess whether a software product with a publicly disclosed vulnerability is installed on any endpoint within the network.

[7.3.](#) Software Management Process Operations

The following operations are necessary to carry out activities within the Software Management domain:

1. Define Guidance: Add software to or remove software from one of three lists for an endpoint. Those lists are software allowed to be installed, software prohibited from being installed, and software that is mandatory for installation.
2. Collect Inventory: Prepare and deliver a "complete" or "update" inventory report to one or more interested inventory data

consumers, or respond to an ad hoc request for inventory data about one or more software products.

3. Evaluate Software Inventory Posture: Based on guidance, evaluate the current software inventory and determine compliance with applicable security policies or identify conditions of interest.
4. Report Evaluation Results: Based on Guidance, report evaluation results to interested report consumers.

[7.4.](#) Information Model Requirements

In this section we describe the data that enterprises will need to carry out each Software Management operation.

[7.4.1.](#) Define Guidance

The "Define Guidance" operation involves the Software Inventory Collection and Evaluation Guidance Capabilities.

The Collection Guidance Capability generates or maintains the guidance related to when Software Inventory should be collected (e.g., periodic or when the inventory changes on an endpoint) and what type of collection (partial or full) should occur at that time.

The Evaluation Guidance Capability generates or maintains the guidance associated with software items where each item is categorized as one of "mandatory", "optional", or "prohibited" for a set of endpoints. A product is "mandatory" if it must be installed on every compatible endpoint. A product is "optional" if it is allowed to be installed on compatible endpoints. A product is "prohibited" if it must not be installed on any compatible endpoints.

The Collection and Evaluation Guidance Capabilities have the following information needs:

- o (M) Unique Software Identifier: the software product which is the subject of the request must be identified.
- o (M) Authorization Status: the authorization status of the product (one of 'mandatory', 'optional', or 'prohibited') must be

supplied.

- o (0) Software Footprint: hashes of the software footprint (or a pointer to those values) may be used to determine if software is corrupted or tampered with.

This operation results in a change to the Collection or Evaluation Guidance. This may, but need not, trigger an automatic enterprise-wide assessment.

[7.4.2.](#) Collect Inventory

The "Collect Inventory" operation involves the following architecture components:

- o Software Inventory Collection Capability
- o Software Inventory Collection Producers and Consumers
- o Software Inventory Collection Guidance Capability

The Software Inventory Collection Capability has the following information needs:

- o (M) Collection Guidance: the requester must indicate when to perform a collection (e.g., at a set time, in response to a change in inventory) and provide any other relevant guidance.
- o (M) Request Type: the requester must indicate whether a "complete" or "update" inventory should be performed and transmitted.
- o (M) Unique Endpoint Identifier: the endpoint whose software inventory data is to be collected must be identified.
- o (M) Endpoint Software Inventory: a description of the current list of software on the endpoint must be supplied.

At the completion of the Collect Inventory operation the Software Inventory Producer will send an enumeration of installed software to the appropriate Software Inventory Collection Consumer(s) or

Repositories.

[7.4.3.](#) Evaluate Software Inventory Posture

The "Evaluate Software Inventory Posture" operation involves the following architecture components:

- o Software Inventory Evaluation Capability
- o Software Inventory Collection Consumers
- o Software Inventory Evaluation Producers and Consumers
- o Software Inventory Evaluation Guidance Capability

The Software Inventory Evaluation Capability is the component which compares endpoint inventory information to current security guidance, and notes any deviations from what is expected.

The Software Inventory Evaluation Capability has the following information needs:

- o (M) Endpoint Identifier: the endpoint whose inventory posture is to be assessed must be identified.
- o (M) Endpoint Software Inventory: a description of the current software inventory of the endpoint must be supplied.

- o (M) Software Inventory Evaluation Guidance: all guidance pertinent to performing an evaluation or assessment of software inventory must be supplied.
- o (M) Software Inventory Evaluation Results: the results from the evaluation of the software inventory against the guidance.

The outcome of this operation is that the Software Inventory Evaluation Capability identifies any deviations from guidance related to the current inventory of software products installed on the endpoint.

7.4.4. Report Evaluation Results

The "Report Evaluation Results" operation involves the following architecture component:

- o Software Inventory Results Report Capability
- o Software Inventory Results Producers and Consumers
- o Software Inventory Evaluation Consumer
- o Software Inventory Reporting Guidance Capability

The Software Inventory Results Report Capability has the following information needs:

- o (M) Endpoint Identifier: the endpoint whose inventory posture assessment is to be reported must be identified.
- o (M) Software Inventory Reporting Guidance: all guidance pertinent to generating and reporting software inventory assessment results must be supplied.
- o (M) Software Inventory Evaluation Results: the results from the evaluation of the software inventory must be supplied.
- o (M) Software Inventory Results Report: the report generated by applying the reporting guidance to the evaluation results.

8. Configuration Management

This section presents an information model for the collection of software configuration posture attributes. Software configuration collection encompasses the subset of tasks within security automation and continuous monitoring involved in the collection of important settings from an endpoint and transmitting those settings to posture

attribute data consumers that store the data and/or perform enterprise-level security posture assessments.

For example, an operating system may enforce configurable password complexity policies. As part of assessing the current complexity

requirements of an network of endpoints using this operating system, a SACM tool will interact with the operating systems using a standardized protocol to retrieve the value of the minimum password length setting. This tool will then verify that each endpoint's minimum password length setting meets (or potentially exceeds) the organizational requirement, and will then report inconsistent endpoints to the responsible administrator for action.

8.1. Core Information Needs

Unique Endpoint Identifier: We need to be able to relate the posture attribute data and assessment results with an endpoint.

Configuration Item Identifier: We need to be able to uniquely identify high-level, cross-platform, configuration statements that can be interpreted and mapped to low-level, platform-specific configuration settings by primary source vendors for their platforms.

Platform Configuration Item Identifier(s): We need to know what low-level configuration items map to the high-level configuration items in order to collect posture attribute data from specific platforms.

Posture Attributes: We need to be able to represent posture attribute data collected from an endpoint for use in assessments.

8.2. Process Area Description

In addition to the compilation of endpoint inventory data, there is a need to compile and assess posture attribute values from an endpoint to ensure that software on an endpoint has been configured correctly. The configuration management security automation domain encompasses a wide range of information needs to define, collect, and evaluate posture attributes across a myriad of operating systems, applications, and endpoint types. Configuration Management requires that there is guidance about how and what to collect. The establishment of all of this guidance is something that needs to be done before the assessment trigger event, and needs to be done in a way that can scale and be sustained over the lifecycle of the applicable software products.

8.2.1. The Existence of Configuration Item Guidance

The model for software configuration collection relies on two main components: (1) the identification of software configuration items and (2) the representation of posture attribute data from an endpoint. For the identification of software configuration items, the primary objective is for the security community to develop a high-level, cross-platform identifier known as a "configuration item" that can then be used by primary source vendors to map to low-level configuration settings called "platform configuration items" for their software. The benefits of this are that a single organization is not responsible for maintaining the entire set of configuration items for all platforms and the primary source vendors are given the flexibility to determine what a particular configuration item means for their software. From a practical perspective, this will likely require a set of federated registries for both "configuration items" and "platform configuration items". An example is the "configuration item" that code should not automatically run when introduced into a system without some entity intentionally invoking that operation. One associated "platform configuration item" for Windows could be disable autorun.

With regards to the model that represents posture attribute data from an endpoint, there are three components: (1) the linking of posture attribute data to the specific endpoint from which it was collected, (2) a generic posture attribute that can be extended by primary source vendors, and (3) the actual extensions of this generic posture attribute by primary source vendors for their platforms. The first component known as "collected posture attributes" associates the posture attribute data collected from an endpoint with that endpoint through the use of the Unique Endpoint Identifier previously mentioned in this model. It may also include other metadata about the collection such as the timestamp and what Posture Attribute Producer was used. This model can be used as the payload for messages of standardized protocols that are responsible for transmitting and receiving posture attribute data between Posture Attribute Producers and Consumers. The second component is the "posture attribute" which provides metadata common to all posture attributes regardless of how they are extended to meet the needs of a particular platform. Finally, the third component is the "platform posture attribute" which is left as an extension point for primary source vendors to fulfill the posture attribute data needs for their platforms. With this model, they can not only define the structure of the posture attribute value data and the data type, but they can also specify any additional metadata about the posture attribute that they feel is relevant for posture attribute data consumers.

[8.2.2.](#) Configuration Collection Guidance

Collection guidance provides instructions on how to collect posture attributes from an endpoint and may include information such as a list of posture attributes that may be collected, a list of posture attributes to collect, and metadata that provides details on when and how to collect posture attributes on an endpoint. The development of this guidance is best performed by the primary source vendor who is the most authoritative source of information for a specific platform. However, if necessary, organizations can translate that information into actionable collection guidance.

[8.2.3.](#) Configuration Evaluation Guidance

With the ability to identify and collect configuration items on an endpoint, the next logical step is to assess the collected posture attribute data against some known state or to check for some specific conditions of interest. This will require the creation of evaluation guidance. Evaluation guidance provides instructions on how to evaluate collected posture attributes from an endpoint against a known state or condition of interest. These instructions will express policies and conditions of interest using logical constructs, requirements for the data used during evaluation (age of data, source of data, etc.), and references to human-oriented data that provides technical, organizational, and other policy information.

The evaluation guidance will need to capture what posture attributes to collect, how to collect the posture attributes (e.g., retrieve it from a configuration database, a published source, or collect it by leveraging collection guidance), any requirements on the usability of the posture attribute data, and instructions on how to evaluate the collected posture attribute data against defined policies or check for a condition of interest. It may also include instructions on the granularity of the results.

[8.2.4.](#) Local Configuration Management Process

Once an endpoint has been targeted for assessment, the first step involves understanding what collection and evaluation guidance is applicable to the endpoint.

After the applicable guidance has been retrieved, the collection of

posture attributes can begin. This operation may result in the collection of a subset of posture attributes on the endpoint or all of the posture attributes on the endpoint using a variety of collection mechanisms.

The three primary collection mechanisms include: (1) retrieving posture attribute data from a enterprise repository, (2) software publishing their posture attribute data, and (3) collecting the posture attribute data from the endpoint or some intermediary. When executing collection guidance, it may be necessary to use some combination of these mechanisms to get all of the required data and it may be necessary to authenticate with the endpoint, CMDB, or some other data store.

In this model, a posture attribute producer may compile and transmit posture attribute data to a posture attribute consumer on any of these occasions: (1) upon request by a posture attribute consumer; (2) on a predefined schedule (e.g. daily, weekly, etc.); or (3) after some event has occurred (e.g. posture attribute value change detected, etc.).

Once the posture attributes have been collected, evaluation guidance is used to assess how the collected posture attribute data compares with the predefined policies or whether or not the endpoint contains conditions of interest. This operation may occur locally on the endpoint or as part of an application that interacts with an intermediate or back end server.

Once completed, the results of the evaluation can be transmitted to the designated locations with the appropriate level of granularity (e.g. results for individual "platform configuration items" or rolled up results for a "configuration item", etc.). Depending on the sensitivity of the evaluation results and collected posture attribute data, it may be necessary for locations receiving the results to authenticate with the sending endpoint and potentially even utilize a secure communication channel.

Last in the configuration management lifecycle is the "adjudicate further action" operation where the results are processed and it is determined if follow up actions are necessary and by which parties.

Further actions could include modifying configuration settings, expelling or quarantining an endpoint from the enterprise network, removing or upgrading installed software, generating an alert, documenting a compliance deviation, or performing some mitigation among other things. This may also require re-initiating the assessment process to ensure follow up actions were completed successfully.

By clearly marking the line between collection and evaluation, tools are free to implement these steps in the way that best suits the needs of the end users and that allows for flexibility and scalability across organizations of all size and shape.

This model allows software and hardware vendors to publish posture attributes in both proactive and reactive manners to centralized repositories for evaluation. This type of flexibility is crucial for scalable security automation in a large and diverse enterprise environment. Finally, with this described model, key stakeholders will be able to quickly and dynamically construct and execute new or updated policy to enable fast and accurate posture evaluation.

Further, evaluation need not be constrained to a single repository of information (be it an endpoint or a central repository). Evaluation can occur across multiple repositories of information to reach an aggregated decision on security posture.

[8.3.](#) Configuration Management Operations

The Configuration Management security automation domain includes all of the processes involved in monitoring the configuration of endpoints on an enterprise network. We have defined the following operations within the configuration management domain:

1. **Define Guidance:** Define or acquire cross-platform configuration item guidance, platform-specific configuration item guidance, collection guidance, and evaluation guidance as applicable for the endpoints that need to be assessed. This may also include verifying the integrity of the guidance.
2. **Collect Posture Attributes:** Gather the needed posture attributes from the endpoint and report them to one or more interested

posture attribute consumers. The collection of posture attributes can be initiated by a number of triggers and can be gathered using a variety of collection mechanisms.

3. Evaluate Posture Attributes: Based on guidance, assess the collected posture attributes from the endpoint to determine compliance with applicable security policies or identify conditions of interest.
4. Report Evaluation Results: Based on guidance, transmit the assessment results to interested report consumers with the appropriate level of granularity.

[8.4.](#) Information Model Requirements

[8.4.1.](#) Define Guidance

The "Define Guidance" operation relies on the existence and population of three types of guidance data: (1) configuration item

guidance (cross-platform guidance and platform-specific guidance), (2) collection guidance, and (3) evaluation guidance.

Therefore, the "Define Guidance" operation involves the Configuration Item, Collection, Evaluation, and Reporting Guidance Capabilities architecture components.

These components generate or store information about configuration items and posture attributes, including when and how to collect them. They also include how to evaluate the collected attributes and rules around reporting the results for the desired configuration posture assessments on applicable endpoints. Configuration Guidance Capabilities can initiate requests to acquire guidance from existing data stores or have the information manually added, modified, or deleted.

To express configuration item guidance, the following information is needed:

- o (M) Configuration Item Identifier: A persistent, unique identifier for the configuration item

- o (M) Configuration Item Description: A high-level description of the configuration item

To express platform configuration item guidance, the following information may be needed ('M' indicates 'mandatory' and 'O' indicates 'optional'):

- o (M) Platform Configuration Item Identifier: A persistent, unique identifier assigned by the primary source vendor
- o (O) A reference to the unique, persistent configuration item identifier
- o (M) Platform Configuration Item Identifier Description: A low-level description of the configuration item for the specific platform
- o (M) Posture Attributes: A list of posture attributes that correspond to the platform configuration item
- o (M) References that provide additional details about the platform configuration item
- o (O) Additional metadata that the primary source vendor feels is relevant

- o (O) References to collection and evaluation guidance that they may have developed or that someone else has developed on their behalf

To express "collection" guidance, the following information is needed:

- o (M) Listings of posture attribute identifiers for which values may be collected and evaluated
- o (M) Lists of attributes that are to be collected
- o (O) Metadata that includes when to collect attributes (e.g. based on interval, event, duration of collection), how and where to collect the posture attribute data (e.g. CMDB, publish, collect

from endpoint or other data source, etc.)

To express "evaluation" guidance, the following information is needed:

- o (M) Logical constructs to express policies and conditions of interest as well as the ability to ask different questions such as "what is the value?", "is a configuration item compliant with a policy?", etc.
- o (M) Data requirements including the age of the data and the source of the data among other things
- o (O) References to human-oriented data that provides technical, organizational, and other policy information

This operation results in a change in posture assessment guidance and may, but need not, trigger an automatic enterprise-wide assessment.

[8.4.2.](#) Collect Posture Attributes Operation

The "Collect Posture Attributes" operation involves the following architectural components:

- o Configuration Collection Producers and Consumers
- o Configuration Collection Capability
- o Configuration Collection Guidance Capability

The Configuration Collection Capability has the following information needs:

- o (M) Collection Guidance: the requester must indicate when to perform a collection (e.g., at a set time, in response to a change in inventory) and provide any other relevant guidance. This guidance may include information such as what posture attributes to collect, how to collect the posture attributes, and whether or not the posture attribute data should be persisted for later use.

- o (M) Request Type: the requester must indicate whether a "full" or "partial" posture attribute collection should be performed.
- o (M) Unique Endpoint Identifier: the endpoint whose posture attribute data is to be collected must be identified.
- o (M) Collected Posture Attributes: the collected posture attribute data to include any required metadata.

At the completion of the Collect Posture Attributes operation the Configuration Collection Producer will send the posture attributes and their values to the appropriate Software Inventory Consumer(s) or Repositories.

[8.4.3.](#) Evaluate Posture Attributes Operation

The "Evaluate Posture Attributes" operation involves the following information elements:

- o Configuration Evaluation Capability
- o Configuration Collection Consumers
- o Configuration Evaluation Producers and Consumers>
- o Configuration Evaluation Guidance Capability

The Configuration Evaluation Capability is the component which compares collected posture attribute data to current evaluation guidance, and notes any deviations.

The Configuration Evaluation Capability has the following information needs:

- o (M) Unique Endpoint Identifier: the endpoint whose inventory posture is to be assessed must be identified.
- o (M) Collected Posture Attributes: the collected posture attribute data to assess must be supplied or retrieved prior to performing the assessment.

- o (M) Configuration Evaluation Guidance: all guidance pertinent to performing an evaluation of posture attribute data must be supplied.
- o (M) Configuration Evaluation Results: the results from the evaluation of the collected posture attributes against the guidance.

8.4.4. Report Evaluation Results Operation

The "Report Evaluation Results" operation involves the following architecture components:

- o Configuration Results Report Capability
- o Configuration Results Producers and Consumers
- o Configuration Evaluation Consumers
- o Configuration Reporting Guidance Capability

The Configuration Results Report Capability has the following information needs:

- o (M) Unique Endpoint Identifier: the endpoint whose posture attribute evaluation results are to be reported must be identified.
- o (M) Posture Assessment Reporting Guidance: all guidance pertinent to generating and reporting posture attribute evaluation results must be supplied. This includes information such as the level of granularity provided within the report (e.g. rolled up to the "configuration item" level, at the "platform configuration level", raw posture attribute data, or the results of evaluating the posture attribute data against a known state), assurance of the evaluation results, etc.
- o (M) Configuration Evaluation Results: the results from the evaluation of the collected posture attributes against the guidance.
- o (M) Configuration Results Report: the report generated by applying the reporting guidance to the evaluation results.

The outcome of this operation is that the Configuration Results Producer reports posture assessment results to interested Configuration Results Consumers.

[9.](#) Vulnerability Management

This section presents an information model for discovering extant vulnerabilities within an enterprise network due to the presence of installed software with publicly disclosed vulnerabilities. Successful vulnerability management builds on the foundation laid by endpoint management, software management, and configuration management, discussed earlier. We limit the scope of vulnerability management to the identification of vulnerable software. We do not currently consider mitigation or remediation of identified vulnerabilities within scope; these are important topics that deserve careful attention. Furthermore, we recognize that the mere presence of installed software with publicly disclosed vulnerabilities does not necessarily mean that an enterprise network is vulnerable to attack, as other defensive layers may effectively preclude exploits. This will also need to be considered within the scope of an information model that supports mitigation/remediation operations.

[9.1.](#) Core Information Needs

Unique Endpoint Identifiers: Each endpoint within the enterprise network must have a unique identifier so we can relate instances of vulnerable software to the endpoint(s) on which they are installed.

Unique Software Identifiers: Organizations need to be able to uniquely identify and label each software product installed on an endpoint. This label must identify software to the version/patch level and must have enough fidelity to be able to associate it with other authoritative data (e.g., a listing of the hashes of the executables that are associated with the software).

Unique Vulnerability Identifiers: Organizations need to be able to uniquely identify and label publicly disclosed software vulnerabilities, and associate those labels with the unique software identifiers of the software product(s) containing those vulnerabilities.

[9.2.](#) Process Area Description

The authors envisage that software publishers produce "vulnerability reports" about their products. They also envisage that each

"vulnerability report" is disseminated in a standard format suitable for machine consumption and processing. Each vulnerability report shall include, at a minimum, a unique identifier of the vulnerability, and a machine-readable "applicability statement" which associates the vulnerability with one or more software products and

any pertinent product configuration information. One day, it will be possible for a machine to authenticate the source of a vulnerability report and confirm that the report has not been tampered with.

Each enterprise network shall support automated processes to recognize when new vulnerability reports are available from software publishers, and to retrieve and process those reports within the context of the enterprise network.

The outcome of vulnerability management processes is the generation of reports and/or alerts that identify vulnerable endpoints.

[9.3.](#) Vulnerability Management Process Operations

We have identified the following operations necessary to carry out activities within the Vulnerability Management domain:

1. **Collect Vulnerability Reports:** Retrieve newly-published vulnerability reports from software publishers.
2. **Evaluate Vulnerability Posture:** Based on guidance, assess the current software inventory and configuration data and determine compliance with applicable security policies or identify conditions of interest.
3. **Report Evaluation Results:** Based on guidance, report evaluation results to interested report consumers.

[9.4.](#) Information Model Requirements

In this section we describe the data that enterprises will need to carry out each Vulnerability Management operation.

[9.4.1.](#) Collect Vulnerability Reports

The "Collect Vulnerability Reports" operation involves the following

architectural components:

- o Vulnerability Collection Guidance Capability
- o Vulnerability Evaluation Guidance Capability

The Vulnerability Collection Guidance Capability maintains a repository of the guidance associated with what information and attributes must be either collected or can be reused from other posture assessment collection operations.

The Vulnerability Evaluation Guidance Capability creates the evaluation guidance associated with individual vulnerability reports. Each vulnerability report documents a publicly disclosed software vulnerability, and associates a unique vulnerability identifier with one or more unique identifiers of affected software products.

The Vulnerability Evaluation Guidance Capability may be maintained by a software publisher or it may be a repository that aggregates vulnerability reports across multiple software publishers.

If it is an aggregate repository, it will also maintain a list of publishers of vulnerability reports.

The Vulnerability Evaluation Guidance Capability may operate in either (or both) "push" or "pull" modes. In "push" mode, publishers of vulnerability reports initiate contact whenever a new report becomes available. In "pull" mode, the Vulnerability Evaluation Guidance Capability initiates contact with publishers, either on a scheduled basis or in response to a request originating within the enterprise network.

The Vulnerability Evaluation Guidance Capability has the following information needs:

- o (0) Publisher List: a new list of publishers may be supplied.
- o (0) Update Schedule: a new schedule for pull-mode operations may be supplied.

This operation may result in a change to the enterprise's collection of vulnerability reports. This may, but need not, trigger an automatic enterprise-wide vulnerability posture assessment.

[9.4.2.](#) Evaluate Vulnerability Posture

The "Evaluate Vulnerability Posture" operation involves the following architectural components:

- o Vulnerability Evaluation Capability
- o Vulnerability Collection Guidance Capability
- o Vulnerability Evaluation Guidance Capability
- o Software Inventory Collection Producers
- o Configuration Evaluation or Reporting Producers (when appropriate)

- o Vulnerability Evaluation Consumers

The Vulnerability Evaluation Capability is the component which compares information about publicly disclosed software vulnerabilities with current information about software installed within the enterprise network. When a vulnerability can be mitigated by a particular configuration, evaluation and/or reporting results can be used to determine the vulnerability posture of the endpoint.

A vulnerability posture assessment may be triggered in response to any of (a) the arrival of a new vulnerability report, (b) a change in software inventory on any enterprise endpoint, or (c) a change in the configuration of any software product installed on any enterprise endpoint. This information is managed by the Vulnerability Collection Guidance Capability.

The Vulnerability Evaluation Capability has the following information needs:

- o (M) Unique Vulnerability Identifier(s): the unique identifier of the vulnerability to be reported on must be supplied.

- o (M) Unique Endpoint Identifiers: the endpoints being assessed for vulnerabilities must be identified.
- o (M) Vulnerability Collection Guidance: all guidance pertinent to determining what previously collected posture data to use must be supplied.
- o (M) Endpoint Software Inventory: a description of the current software inventory of the endpoint must be supplied.
- o (M) Configuration Evaluation Results: the results from the evaluation of the collected posture attributes against the guidance. OR (M) Configuration Results Report: the report generated by applying the reporting guidance to the evaluation results. Configuration Posture Assessment Results would only be required when the vulnerability can be mitigated by configuring an installed instance of software in a particular manner.
- o (M) Vulnerability Evaluation Guidance: all guidance pertinent to performing an evaluation of posture data must be supplied.
- o (M) Vulnerability Evaluation Results: the results from the evaluation of the collected posture data against the guidance.

The outcome of this operation is that a vulnerability report is delivered to Vulnerability Evaluation Consumers.

[9.4.3.](#) Report Evaluation Results

The "Report Evaluation Results" operation involves the following architectural components:

- o Vulnerability Results Report Capability
- o Vulnerability Results Producers and Consumers
- o Vulnerability Evaluation Consumers
- o Vulnerability Reporting Guidance Capability

The Vulnerability Results Report Capability has the following information needs:

- o (M) Vulnerability Evaluation Results: the results from the evaluation of the collected posture data against the guidance.
- o (M) Unique Endpoint Identifiers: the vulnerable endpoints must be identified.
- o (M) Vulnerability Assessment Reporting Guidance: all guidance pertinent to generating and reporting vulnerability assessment results must be supplied (e.g., an alert should be generated).
- o (M) Vulnerability Results Report: the report generated by applying the reporting guidance to the evaluation results.

The outcome of this operation may be a report, an alert, or set of reports and/or alerts, identifying any vulnerable endpoints within the enterprise network.

10. From Information Needs to Information Elements

The previous sections highlighted information needs for a set of management process areas that use posture assessment to achieve organizational security goals. A single information need may be made up of multiple information elements. Some information elements may be required for two different process areas, resulting in two different requirements. In an effort to support the main idea of collect once and reuse the data to support multiple processes, we try to define a singular set of information elements that will support all the associated information needs.

11. Information Model Elements

Traditionally, one would use the SACM architecture to define interfaces that required information exchanges. Identified information elements would then be based on those exchanges. Because the SACM architecture document is still in the personal draft stage, this information model uses a different approach to the identification of information elements. First it lists the four main

endpoint posture assessment activities. Then it identifies management process areas that use endpoint posture assessment to achieve organizational security objectives. These process areas were then broken down into operations that mirrored the typical workflow from the SACM Use Cases draft [[I-D.ietf-sacm-use-cases](#)]. These operations identify architectural components and their information needs. In this section, information elements derived from those information needs are mapped back to the four main activities listed above.

The original liaison statement [[IM-LIAISON-STATEMENT-NIST](#)] requested contributions for the SACM information model in the four areas described below. Based on the capabilities defined previously in this document, the requested areas alone do not provide a sufficient enough categorization of the necessary information model elements. The following sub-sections directly address the requested areas as follows:

1. Endpoint Identification
 - A. [Section 11.1](#) Asset Identifiers: Describes identification of many different asset types including endpoints.
2. Endpoint Characterization
 - A. [Section 11.3](#) Endpoint characterization: This directly maps to the requested area.
3. Endpoint Attribute Expression/Representation
 - A. [Section 11.4](#) Posture Attribute Expression: This corresponds to the first part of "Endpoint Attribute Expression/Representation."
 - B. [Section 11.5](#) Actual Value Representation: This corresponds to the second part of "Endpoint Attribute Expression/Representation."
4. Policy evaluation expression and results reporting

- A. [Section 11.6](#) Evaluation Guidance: This corresponds to the

first part of "Policy evaluation expression and results reporting."

- B. [Section 11.7](#) Evaluation Result Reporting: corresponds to the second part of "Policy evaluation expression and results reporting."

Additionally, [Section 11.2](#) Other Identifiers: describes other important identification concepts that were not directly requested by the liaison statement.

Per the liaison statement, each subsection references related work that provides a basis for potential data models. Some analysis is also included for each area of related work on how directly applicable the work is to the SACM efforts. In general, much of the related work does not fully address the general or use case-based requirements for SACM, but they do contain some parts that can be used as the basis for data models that correspond to the information model elements. In these cases additional work will be required by the WG to adapt the specification. In some cases, existing work can largely be used in an unmodified fashion. This is also indicated in the analysis. Due to time constraints, the work in this section is very biased to previous work supported by the authors and does not reflect a comprehensive listing. An attempt has been made where possible to reference existing IETF work. Additional research and discussion is needed to include other related work in standards and technology communities that could and should be listed here. The authors intend to continue this work in subsequent revisions of this draft.

Where possible when selecting and developing data models in support of these information model elements, extension points and IANA registries SHOULD be used to provide for extensibility which will allow for future data models to be addressed.

[11.1](#). Asset Identifiers

In this context an "asset" refers to "anything that has value to an organization" (see [[NISTIR-7693](#)]). This use of the term "asset" is broader than the current definition in [[I-D.ietf-sacm-terminology](#)]. To support SACM use cases, a number of different asset types will need to be addressed. For each type of asset, one or more type of asset identifier will be needed for use in establishing contextual relationships within the SACM information model. The following asset types are referenced or implied by the SACM use cases:

Endpoint: Identifies an individual endpoint for which posture is collected and evaluated.

Hardware: Identifies a given type of hardware that may be installed within an endpoint.

Software: Identifies a given type of software that may be installed within an endpoint.

Network: Identifies a network for which a given endpoint may be connected or request a connection to.

Organization: Identifies an organizational unit.

Person: Identifies an individual, often within an organizational context.

[11.1.1.1.](#) Related Work

[11.1.1.1.1.](#) Asset Identification

The Asset Identification specification [[NISTIR-7693](#)] is an XML-based data model that "provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information about the assets." Asset identification plays an important role in an organization's ability to quickly correlate different sets of information about assets. The Asset Identification specification provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information about the assets. Asset Identification provides a relatively flat and extensible model for capturing the identifying information about a one or more assets, and also provides a way to represent relationships between assets.

The model is organized using an inheritance hierarchy of specialized asset types/classes (see Figure 4), providing for extension at any level of abstraction. For a given asset type, a number of properties are defined that provide for capturing identifying characteristics and the referencing of namespace qualified asset identifiers, called "synthetic IDs."

The following figure illustrates the class hierarchy defined by the Asset Identification specification.

```
asset
+-it-asset
| +-circuit
| +-computing-device
| +-database
| +-network
| +-service
| +-software
| +-system
| +-website
+-data
+-organization
+-person
```

Figure 4: Asset Identification Class Hierarchy

This table presents a mapping of notional SACM asset types to those asset types provided by the Asset Identification specification.

SACM Asset Type	Asset Identification Type	Notes
Endpoint	computing-device	This is not a direct mapping since a computing device is not required to have network-connectivity. Extension will be needed to define a directly aligned endpoint asset type.
Hardware	Not Applicable	The concept of hardware is not addressed by the asset identification specification. An extension can be created based on the it-asset class to address this concept.
Software	software	Direct mapping.
Network	network	Direct mapping.
Organization	organization	Direct mapping.
Person	person	Direct mapping.

Table 1: Mapping of SACM to Asset Identification Asset Types

This specification has been adopted by a number of SCAP validated products. It can be used to address asset identification and categorization needs within SACM with minor modification.

[11.1.2.](#) Endpoint Identification

An unique name for an endpoint. This is a foundational piece of information that will enable collected posture attributes to be related to the endpoint from which they were collected. It is important that this name either be created from, provide, or be associated with operational information (e.g., MAC address, hardware certificate) that is discoverable from the endpoint or its communications on the network. It is also important to have a method of endpoint identification that can persist across network sessions to allow for correlation of collected data over time.

[11.1.2.1.](#) Related Work

The previously introduced asset identification specification (see [Section 11.1.1.1](#) provides a basis for endpoint identification using the "computing-device" class. While the meaning of this class is broader than the current definition of an endpoint in the SACM terminology [[I-D.ietf-sacm-terminology](#)], either that class or an appropriate sub-class extension can be used to capture identification information for various endpoint types.

[11.1.3.](#) Software Identification

A unique name for a unit of installable software. Software names should generally represent a unique release or installable version of software. Identification approaches should allow for identification of commercially available, open source, and organizationally developed custom software. As new software releases are created, a new software identifier should be created by the releasing party (e.g., software creator, publisher, licensor). Such an identifier is useful to:

- o Relate metadata that describes the characteristics of the unit of software, potentially stored in a repository of software information. Typically, the software identifier would be used as an index into such a repository.

- o Indicate the presence of the software unit on a given endpoint.
- o To determine what endpoints are the targets for an assessment based on what software is installed on that endpoint.
- o Define guidance related to a software unit that represents collection, evaluation, or other automatable policies.

In general, an extensible method of software identification is needed to provide for adequate coverage and to address legacy identification approaches. Use of an IANA registry supporting multiple software identification methods would be an ideal way forward.

[11.1.3.1.](#) Related Work

While we are not aware of a one-size-fits-all solution for software identification, there are two existing specifications that should be considered as part of the solution set. They are described in the following subsections.

Waltermire & Watson	Expires January 4, 2015	[Page 41]
---------------------	-------------------------	-----------

Internet-Draft	Endpoint Assessment Information Model	July 2014
----------------	---------------------------------------	-----------

[11.1.3.1.1.](#) Common Platform Enumeration

[11.1.3.1.1.1.](#) Background

The Common Platform Enumeration (CPE) [[CPE-WEBSITE](#)] is composed of a family of four specification that are layered to build on lower-level functionality. The following describes each specification:

1. CPE Naming: A standard machine-readable format [[NISTIR-7695](#)] for encoding names of IT products and platforms. This defines the notation used to encode the vendor, software name, edition, version and other related information for each platform or product. With the 2.3 version of CPE, a second, more advanced notation was added to the original colon-delimited notation for CPE naming.
2. CPE Matching: A set of procedures [[NISTIR-7696](#)] for comparing names. This describes how to compare two CPE names to one

another. It describes a logical method that ensures that automated systems comparing two CPE names would arrive at the same conclusion.

3. CPE Applicability Language: An XML-based language [[NISTIR-7698](#)] for constructing "applicability statements" that combine CPE names with simple logical operators.
4. CPE Dictionary: An XML-based catalog format [[NISTIR-7697](#)] that enumerates CPE Names and associated metadata. It details how to encode the information found in a CPE Dictionary, thereby allowing multiple organizations to maintain compatible CPE Dictionaries.

The primary use case of CPE is for exchanging software inventory data, as it allows the usage of unique names to identify software platforms and products present on an endpoint. The NIST currently maintains and updates a dictionary of all agreed upon CPE names, and is responsible for ongoing maintenance of the standard. Many of the names in the CPE dictionary have been provided by vendors and other 3rd-parties.

While the effort has seen wide adoption, most notably within the US Government, a number of critical flaws have been identified. The most critical issues associated with the effort are:

- o Because there is no requirement for vendors to publish their own, official CPE names, CPE necessarily requires one or more organizations for curation. This centralized curation requirement ensures that the effort has difficulty scaling.

- o Not enough primary source vendors provide platform and product naming information. As a result, this pushes too much of the effort out onto third-party groups and non-authoritative organizations. This exacerbates the ambiguity in names used for identical platforms and products and further reduces the utility of the effort.

[11.1.3.1.1.2](#). Applicability to Software Identification

The Common Platform Enumeration (CPE) Naming specification version 2.3 defines a scheme for human-readable standardized identifiers of

hardware and software products.

CPE names are the identifier format for software and hardware products used in SCAP 1.2 and is currently adopted by a number of SCAP product vendors.

CPE names can be directly referenced in the asset identification software class (see [Section 11.1.1.1.](#))

Although relevant, CPE has an unsustainable maintenance "tail" due to the need for centralized curation and naming-consistency enforcement. Its mention in this document is to support the historic inclusion of CPE as part of SCAP and implementation of this specification in a number of security processes and products. Going forward, software identification (SWID) tags are recommended as a replacement for CPE. To this end, work has been started to align both efforts to provide translation for software units identified using SWID tags to CPE Names. This translation would allow tools that currently use CPE-based identifiers to map to SWID identifiers during a transition period.

[11.1.3.1.2.](#) Software Identification (SWID) Tags

The software identification tag specification [[ISO.19770-2](#)] is an XML-based data model that is used to describe a unit of installable software. A SWID tag contains data elements that:

- o Identify a specific unit of installable software,
- o Enable categorization of the software (e.g., edition, bundle),
- o Identification and hashing of software artifacts (e.g., executables, shared libraries),
- o References to related software and dependencies, and
- o Inclusion of extensible metadata.

SWID tags can be associated with software installation media, installed software, software updates (e.g., service packs, patches, hotfixes), and redistributable components. SWID tags also provide for a mechanism to relate these concepts to each other. For example,

installed software can be related back to the original installation media, patches can be related to the software that they patch, and software dependencies can be described for required redistributable components. SWID tags are ideally created at build-time by the software creator, publisher or licensor; are bundled with software installers; and are deployed to an endpoint during software installation.

SWID tags should be considered for two primary uses:

1. As the data format for exchanging descriptive information about software products, and
2. As the source of unique identifiers for installed software.

In addition to usage for software identification, a SWID tag can provide the necessary data needed to target guidance based on included metadata, and to support verification of installed software and software media using cryptographic hashes. This added information increases the value of using SWID tags as part of the larger security automation and continuous monitoring solution space.

11.1.4. Hardware Identification

Due to the time constraints, research into information elements and related work for identifying hardware is not included in this revision of the information model.

11.2. Other Identifiers

In addition to identifying core asset types, it is also necessary to have stable, globally unique identifiers to represent other core concepts pertaining to posture attribute collection and evaluation. The concept of "global uniqueness" ensures that identifiers provided by multiple organization do not collide. This may be handled by a number of different mechanisms (e.g., use of namespaces).

11.2.1. Platform Configuration Item Identifier

A name for a low-level, platform-dependent configuration mechanism as determined by the authoritative primary source vendor. New identifiers will be created when the source vendor makes changes to the underlying platform capabilities (e.g., adding new settings, replacing old settings with new settings). When created each

identifier should remain consistent with regards to what it represents. Generally, a change in meaning would constitute the creation of a new identifier.

For example, if the configuration item is for "automatic execution of code", then the platform vendor would name the low-level mechanism for their platform (e.g., autorun for mounted media).

[11.2.1.1](#). Related Work

[11.2.1.1.1](#). Common Configuration Enumeration

The Common Configuration Enumeration (CCE) [[CCE](#)] is an effort managed by NIST. CCE provides a unique identifier for platform-specific configuration items that facilitates fast and accurate correlation of configuration items across multiple information sources and tools. CCE does this by providing an identifier, a human readable description of the configuration control, parameters needed to implement the configuration control, various technical mechanisms that can be used to implement the configuration control, and references to documentation that describe the configuration control in more detail.

By vendor request, NIST issues new blocks of CCE identifiers. Vendors then populate the required fields and provided the details back to NIST for publication in the "CCE List", a consolidated listing of assigned CCE identifiers and associated data. Many vendors also include references to these identifiers in web pages, SCAP content, and prose configuration guides they produce.

CCE the identifier format for platform specific configuration items in SCAP and is currently adopted by a number of SCAP product vendors.

While CCE is largely supported as a crowd-sourced effort, it does rely on a central point of coordination for assignment of new CCE identifiers. This approach to assignment requires a single organization, currently NIST, to manage allocations of CCE identifiers which doesn't scale well and introduces sustainability challenges for large volumes of identifier assignment. If this approach is used going forward by SACM, a namespaced approach is recommended for identifier assignment that allows vendors to manage their own namespace of CCE identifiers. This change would require additional work to specify and implement.

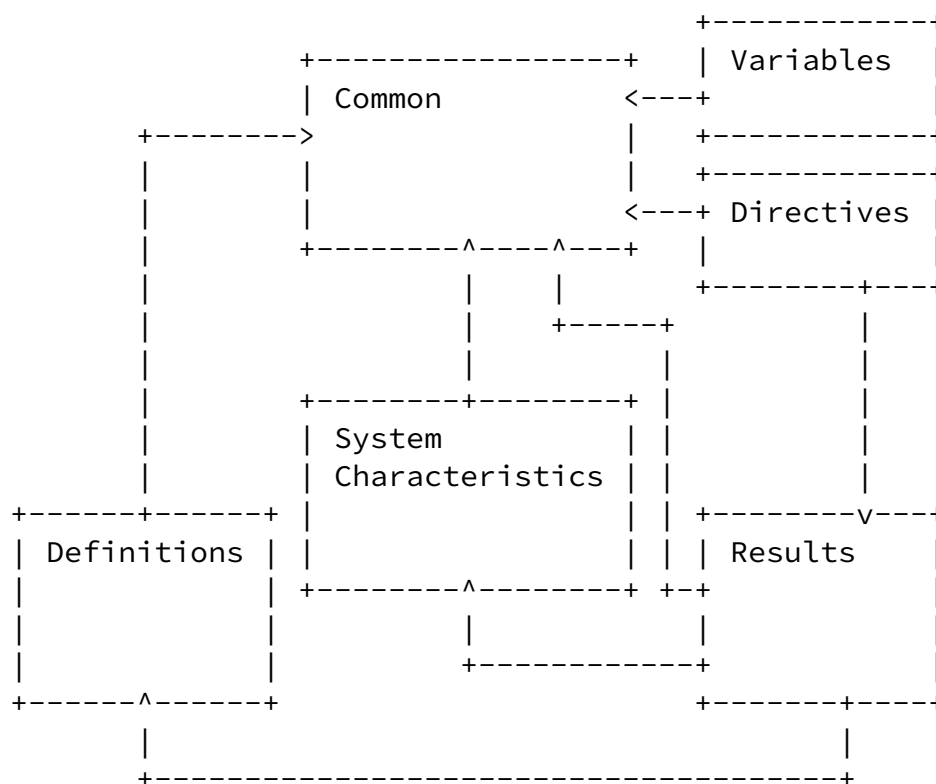
[11.2.1.1.2.](#) Open Vulnerability and Assessment Language[11.2.1.1.2.1.](#) Background

The Open Vulnerability and Assessment Language (OVAL(R)) is an XML schema-based data model developed as part of a public-private information security community effort to standardize how to assess and report upon the security posture of endpoints. OVAL provides an established framework for making assertions about an endpoint's posture by standardizing the three main steps of the assessment process:

1. representing the current endpoint posture;
2. analyzing the endpoint for the presence of the specified posture;
and
3. representing the results of the assessment.

OVAL facilitates collaboration and information sharing among the information security community and interoperability among tools. OVAL is used internationally and has been implemented by a number of operating system and security tools vendors.

The following figure illustrates the OVAL data model.



Note: The direction of the arrows indicate a model dependency

Figure 5: The OVAL Data Model

The OVAL data model [[OVAL-LANGUAGE](#)], visualized in Figure 5, is composed of a number of different components. The components are:

- o Common: Constructs, enumerations, and identifier formats that are used throughout the other model components.

- o Definitions: Constructs that describe assertions about system state. This component also includes constructs for internal variable creation and manipulation through a variety of functions. The core elements are:
 - * Definition: A collection of logical statements that are combined to form an assertion based on endpoint state.
 - * Test(platform specific): A generalized construct that is extended in platform schema to describe the evaluation of expected against actual state.

- * Object(platform specific): A generalized construct that is extended in platform schema to describe a collectable aspect of endpoint posture.
- * State(platform specific): A generalized construct that is extended in platform schema to describe a set of criteria for evaluating posture attributes.
- o Variables: Constructs that allow for the parameterization of the elements used in the Definitions component based on externally provided values.
- o System Characteristics: Constructs that represent collected posture from one or more endpoints. This element may be embedded with the Results component, or may be exchanged separately to allow for separate collection and evaluation. The core elements of this component are:
 - * CollectedObject: Provides a mapping of collected Items to Objects defined in the Definitions component.
 - * Item(platform specific): A generalized construct that is extended in platform schema to describe specific posture attributes pertaining to an aspect of endpoint state.
- o Results: Constructs that represent the result of evaluating

expected state (state elements) against actual state (item elements). It includes the true/false evaluation result for each evaluated Definition and Test. Systems characteristics are embedded as well to provide low-level posture details.

- o Directives: Constructs that enable result reporting detail to be declared, allowing for result production to be customized.

End-user organizations and vendors create assessment guidance using OVAL by creating XML instances based on the XML schema implementation of the OVAL Definitions model. The OVAL Definitions model defines a structured identifier format for each of the Definition, Test, Object, State, and Item elements. Each instantiation of these elements in OVAL XML instances are assigned a unique identifier based on the specific elements identifier syntax. These XML instances are used by tools that support OVAL to drive collection and evaluation of endpoint posture. When posture collection is performed, an OVAL Systems Characteristics XML instance is generated based on the collected posture attributes. When this collected posture is evaluated, an OVAL Result XML instance is generated that contains the results of the evaluation. In most implementations, the collection and evaluation is performed at the same time.

Many of the elements in the OVAL model (i.e., Test, Object, State, Item) are abstract, requiring a platform-specific schema implementation, called a "Component Model" in OVAL. These platform schema implementations are where platform specific posture attributes are defined. For each aspect of platform posture a specialized OVAL Object, which appears in the OVAL Definitions model, provides a format for expressing what posture attribute data to collect from an endpoint through the specification of a datatype, operation, and value(s) on entities that uniquely identify a platform configuration item. For example, a hive, key, and name is used to identify a registry key on a Windows endpoint. Each specialized OVAL Object has a corresponding specialized State, which represents the posture attributes that can be evaluated, and an Item which represents the specific posture attributes that can be collected. Additionally, a specialized Test exists that allows collected Items corresponding to a CollectedObject to be evaluated against one or more specialized States of the same posture type.

The OVAL language provides a generalized approach suitable for

posture collection and evaluation. While this approach does provide for a degree of extensibility, there are some concerns that should be addressed in order to make OVAL a viable basis for SACM's use. These concerns include:

- o Platform Schema Creation and Maintenance: In OVAL platform schema, the OVAL data model maintains a tight binding between the Test, Object, State, and Item elements used to assess an aspect of endpoint posture. Creating a new platform schema or adding a new posture aspect to an existing platform schema can be a very labor intensive process. Doing so often involves researching and understanding system APIs and can be prone to issues with inconsistency within and between platforms. To simplify platform schema creation and maintenance, the model needs to be evolved to generalize the Test, Object, and State elements, requiring only the definition of an Item representation.
- o Given an XML instance based on the Definitions model, it is not clear in the specification how incremental collection and evaluation can occur. Because of this, typically, OVAL assessments are performed on a periodic basis. The OVAL specification needs to be enhanced to include specifications for performing event-based and incremental assessment in addition to full periodic collection.
- o Defining new functions for manipulating variable values is current handled in the Definitions schema. This requires revision to the core language to add new functions. The OVAL specification needs

to be evolved to provide for greater extensibility in this area, allowing extension schema to define new functions.

- o The current process for releasing a new version of OVAL, bundle releases of the core language with release of community recognized platform schema. The revision processes for the core and platform schema need to be decoupled. Each platform schema should use some mechanism to declare which core language version it relies on.

If adopted by SCAM, these issues will need to be addressed as part of the SCAM engineering work to make OVAL more broadly adoptable as a general purpose data model for posture collection and evaluation.

11.2.1.1.2.2. Applicability to Platform Configuration Item Identification

Each OVAL Object is identified by a globally unique identifier. This globally unique identifier could be used by the SACM community to identify platform-specific configuration items and at the same time serve as collection guidance. If used in this manner, OVAL Objects would likely need to undergo changes in order to decouple it from evaluation guidance and to provide more robust collection capabilities to support the needs of the SACM community.

11.2.2. Configuration Item Identifier

An identifier for a high-level, platform-independent configuration control. This identification concept is necessary to allow similar configuration item concepts to be comparable across platforms. For example, a configuration item might be created for the minimum password length configuration control, which may then have a number of different platform-specific configuration settings. Without this type of identification, it will be difficult to perform evaluation of expected versus actual state in a platform-neutral way.

High-level configuration items tend to change much less frequently than the platform-specific configuration items (see [Section 11.2.1](#)) that might be associated with them. To provide for the greatest amount of sustainability, collections of configuration item identifiers are best defined by specific communities of interest, while platform-specific identifiers are best defined by the source vendor of the platform. Under this model, the primary source vendors would map their platform-specific configuration controls to the appropriate platform-independent item allowing end-user organizations to make use of these relationships.

To support different communities of interest, it may be necessary to support multiple methods for identification of configuration items

and for associating related metadata. Use of an IANA registry supporting multiple configuration item identification methods would be an ideal way forward. To the extent possible, a few number of configuration item identification approaches is desirable, to maximize the update by vendors who would be maintain mapping of

platform-specific configuration identifiers to the more general platform-neutral configuration identifiers.

[11.2.2.1.](#) Related Work

[11.2.2.1.1.](#) Control Correlation Identifier

The Control Correlation Identifier (CCI) [[CCI](#)] is developed and managed by the United States Department of Defense (US-DoD) Defense Information Systems Agency (DISA). According to their website, CCI "provides a standard identifier and description for each of the singular, actionable statements that comprise an information assurance (IA) control or IA best practice. CCI bridges the gap between high-level policy expressions and low-level technical implementations. CCI allows a security requirement that is expressed in a high-level policy framework to be decomposed and explicitly associated with the low-level security setting(s) that must be assessed to determine compliance with the objectives of that specific security control. This ability to trace security requirements from their origin (e.g., regulations, IA frameworks) to their low-level implementation allows organizations to readily demonstrate compliance to multiple IA compliance frameworks. CCI also provides a means to objectively roll-up and compare related compliance assessment results across disparate technologies."

It is recommended that this approach be analysed as a potential candidate for use as a configuration item identifier method.

Note: This reference to CCI is for informational purposes. Since the editors do not represent DISA's interests, its inclusion in this document does not indicate the presence or lack of desire to contribute aspects of this effort to SACM.

[11.2.2.1.2.](#) A Potential Alternate Approach

There will likely be a desire by different communities to create different collections of configuration item identifiers. This fracturing may be caused by:

- o Different requirements for levels of abstraction,
- o Varying needs for timely maintenance of the collection, and

- o Differing scopes of technological needs.

Due to these and other potential needs, it will be difficult to standardize around a single collection of configuration identifiers. A workable solution will be one that is scalable and usable for a broad population of end-user organizations. An alternate approach that should be considered is the definition of data model that contains a common set of metadata attributes, perhaps supported by an extensible taxonomy, that can be assigned to platform-specific configuration items. If defined at a necessary level of granularity, it may be possible to query collections of platform-specific configuration items provided by vendors to create groupings at various levels of abstractions. By utilizing data provided by vendors, technological needs and the timeliness of information can be addressed based on customer requirements.

SACM should consider this and other approaches to satisfy the need for configuration item roll-up in a way that provides the broadest benefit, while achieving a sensible degree of scalability and sustainability.

[11.2.3.](#) Vulnerability Identifier

An unique name for a known software flaw that exists in specific versions of one or more units of software. One use of a vulnerability identifier in the SACM context is to associate a given flaw with the vulnerable software using software identifiers. For this reason at minimum, software identifiers should identify a software product to the patch or version level, and not just to the level that the product is licensed.

[11.2.3.1.](#) Related Work

[11.2.3.1.1.](#) Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) [[CVE-WEBSITE](#)] is a MITRE led effort to assign common identifiers to publicly known security vulnerabilities in software to facilitate the sharing of information related to the vulnerabilities. CVE is the industry standard by which software vendors, tools, and security professionals identify vulnerabilities and could be used to address SACM's need for a vulnerability identifier.

[11.3.](#) Endpoint characterization

Target when policies (collection, evaluated, guidance) apply

Collection can be used to further characterize

Also human input

Information required to characterize an endpoint is used to determine what endpoints are the target of a posture assessment. It is also used to determine the collection, evaluation, and/or reporting policies and the associated guidance that apply to the assessment. Endpoint characterization information may be populated by:

- o A manual input process and entered into records associated with the endpoint, or
- o Using information collected and evaluated by an assessment.

Regardless of the method of collection, it will be necessary to query and exchange endpoint characterization information as part of the assessment planning workflow.

[11.3.1.](#) Related Work

[11.3.1.1.](#) Extensible Configuration Checklist Description Format

[11.3.1.1.1.](#) Background

The Extensible Configuration Checklist Description Format (XCCDF) is a specification that provides an XML-based format for expressing security checklists. The XCCDF 1.2 specification is published by International Organization for Standardization (ISO) [[ISO.18180](#)]. XCCDF contains multiple components and capabilities, and various components align with different elements of this information model.

This specification was originally published by NIST [[NISTIR-7275](#)]. When contributed to ISO Joint Technical Committee 1 (JTC 1), a comment was introduced indicating an interest in the IETF becoming the maintenance organization for this standard. If the SACM working group is interested in taking on engineering work pertaining to XCCDF, a contribution through a national body can be made to create a ballot resolution for transition of this standard to the IETF for maintenance.

[11.3.1.1.2.](#) Applicability to Endpoint characterization

The target component of XCCDF provides a mechanism for capturing

characteristics about an endpoint including the fully qualified domain name, network address, references to external identification information (e.g. Asset Identification), and is extensible to support other useful information (e.g. MAC address, globally unique identifier, certificate, etc.). XCCDF may serve as a good starting

point for understanding the types of information that should be used to identify an endpoint.

[11.3.1.2.](#) Asset Reporting Format

[11.3.1.2.1.](#) Background

The Asset Reporting Format (ARF) [[NISTIR-7694](#)] is a data model to express information about assets, and the relationships between assets and reports. It facilitates the reporting, correlating, and fusing of asset information within and between organizations. ARF is vendor and technology neutral, flexible, and suited for a wide variety of reporting applications.

There are four major sub-components of ARF:

- o Asset: The asset component element includes asset identification information for one or more assets. It simply houses assets independent of their relationships to reports. The relationship section can then link the report section to specific assets.
- o Report: The report component element contains one or more asset reports. An asset report is composed of content (or a link to content) about one or more assets.
- o Report-Request: The report-request component element contains the asset report requests, which can give context to asset reports captured in the report section. The report-request section simply houses asset report requests independent of the report which was subsequently generated.
- o Relationship: The relationship component element links assets, reports, and report requests together with well-defined relationships. Each relationship is defined as {subject} {predicate} {object}, where {subject} is the asset, report

request, or report of interest, {predicate} is the relationship type being established, and {object} is one or more assets, report requests, or reports.

[11.3.1.2.2.](#) Relationship to Endpoint Characterization

For Endpoint Characterization, ARF can be used in multiple ways due to its flexibility. ARF supports the use of the Asset Identification specification (more in [Section 11.3.1.2.3](#)) to embed the representation of one or more assets as well as relationships between those assets. It also allows the inclusion of report-requests, which can provide details on what data was required for an assessment.

ARF is agnostic to the data formats of the collected posture attributes and therefore can be used within the SACM Architecture to provide Endpoint Characterization without dictating data formats for the encoding of posture attributes. The embedded Asset Identification data model (see [Section 11.1.1.1](#)) can be used to characterize one or more endpoints to allow targeting for collection, evaluation, etc. Additionally, the report-request model can dictate the type of reporting that has been requested, thereby providing context as to which endpoints the guidance applies.

[11.3.1.2.3.](#) Asset Identification

Described earlier

In the context of Endpoint Characterization, the Asset Identification data model could be used to encode information that identifies specific endpoints and/or classes of endpoints to which a particular assessment is relevant. The flexibility in the Asset Identification specification allows usage of various endpoint identifiers as defined by the SACM engineering work.

As stated in [Section 11.3.1.2.3](#), the Asset Identification specification is included within the Asset Reporting Framework (ARF) and therefore can be used in concert with that specification as well.

[11.3.1.3.](#) The CPE Applicability Language

CPE described earlier

Applicability in CPE is defined as an XML language [[NISTIR-7698](#)] for using CPE names to create applicability statements using logical expressions. These expressions can be used to applicability statements that can drive decisions about assets, whether or not to do things like collect data, report data, and execute policy compliance checks.

It is recommended that SACM evolve the CPE Applicability Language through engineering work to allow it to better fit into the security automation vision laid out by the Use Cases and Architecture for SACM. This should include de-coupling the identification part of the language from the logical expressions, making it such that the language is agnostic to the method by which assets are identified. This will allow use of SWID, CPE Names, or other identifiers to be used, perhaps supported by an IANA registry of identifier types.

The other key aspect that should be evolved is the ability to make use of the Applicability Language against a centralized repository of collected posture attributes. The language should be able to make

applicability statements against previously collected posture attributes, such that an enterprise can quickly query the correct set of applicable endpoints in an automated and scalable manner.

[11.4.](#) Posture Attribute Expression

Discuss the catalog concept. Listing of things that can be chosen from. Things we can know about. Vendors define catalogs. Ways for users to get vendor-provided catalogs.

To support the collection of posture attributes, there needs to be a way for operators to identify and select from a set of platform-specific attribute(s) to collect. The same identified attributes will also need to be identified post-collection to associate the actual value of that attribute pertaining to an endpoint as it was configured at the time of the collection. To provide for extensibility, the need exists to support a variety of possible identification approaches. It is also necessary to enable vendors of software to provide a listing, or catalog, of the available posture attributes to operators that can be collected. Ideally, a federated approach will be used to allow organizations to identify the location

for a repository containing catalogs of posture attributes provided by authoritative primary source vendors. By querying these repositories, operators will be able to acquire the appropriate listings of available posture attributes for their deployed assets. One or more posture attribute expressions are needed to support these exchanges.

[11.4.1.](#) Related Work

The ATOM Syndication Format [[RFC4287](#)] provides an extensible, flexible XML-based expression for organizing a collection of data feeds consisting of entries. This standard can be used to express one or more catalogs of posture attributes represented as data feeds. Groupings of posture attributes would be represented as entries. These entries could be defined using the data models described in the "Related Work" sections below. Additionally, this approach can also be used more generally for guidance repositories allowing other forms of security automation guidance to be exchanged using the same format.

[11.4.2.](#) Platform Configuration Attributes

A low-level, platform-dependent posture attribute as determined by the authoritative primary source vendor. Collection guidance will be derived from catalogs of platform specific posture attributes.

For example, a primary source vendor would create a platform-specific posture attribute that best models the posture attribute data for their platform.

[11.4.2.1.](#) Related Work

[11.4.2.1.1.](#) Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in [Section 11.2.1.1.2.1](#). The OVAL System Characteristics platform extension models provide a catalog of the posture attributes that can be collected from an endpoint. In OVAL these posture attributes are further grouped into logical constructs called OVAL Items. For example, the passwordpolicy_item that is part of the Windows platform

extension groups together all the posture attributes related to passwords for an endpoint running Windows (e.g. maximum password age, minimum password length, password complexity, etc.). The various OVAL Items defined in the OVAL System Characteristics may serve as a good starting for the types of posture attribute data that needs to be collected from endpoints.

OVAL platform extension models may be shared using the ATOM Syndication Format.

11.4.2.1.2. Network Configuration Protocol and YANG Data Modeling Language

The Network Configuration Protocol (NETCONF) [[RFC6241](#)] defines a mechanism for managing and retrieving posture attribute data from network infrastructure endpoints. The posture attribute data that can be collected from a network infrastructure endpoint is highly extensible and can be defined using the YANG modeling language [[RFC6020](#)]. Models exist for common datatypes, interfaces, and routing subsystem information among other subjects. The YANG modeling language may be useful in providing an extensible framework for the SACM community to define one or more catalogs of posture attribute data that can be collected from network infrastructure endpoints.

Custom YANG modules may also be shared using the ATOM Syndication Format.

11.4.2.1.3. Simple Network Management Protocol and Management Information Base Entry

The Simple Network Protocol (SNMP) [[RFC3411](#)] defines a protocol for managing and retrieving posture attribute data from endpoints on a network. The posture attribute data that can be collected of an

endpoint and retrieved by SNMP is defined by the Management Information Base (MIB) [[RFC3418](#)] which is hierarchical collection of information that is referenced using Object Identifiers. Given this, MIBs may provide an extensible way for the SACM community to define a catalog of posture attribute data that can be collected off of endpoints using SNMP.

MIBs may be shared using the ATOM Syndication Format.

[11.5.](#) Actual Value Representation

Discuss instance concept.

The actual value of a posture attribute is collected or published from an endpoint. The identifiers discussed previously provide names for the posture attributes (i.e., software or configuration item) that can be the subject of an assessment. The information items listed below are the actual values collected during the assessment and are all associated with a specific endpoint.

[11.5.1.](#) Software Inventory

A software inventory is a list of software identifiers (or content) associated with a specific endpoint. Software inventories are maintained in some organized fashion so that entities can interact with it. Just having software publish identifiers onto an endpoint is not enough, there needs to be an organized listing of all those identifiers associated with that endpoint.

[11.5.1.1.](#) Related Work

[11.5.1.1.1.](#) Asset Summary Reporting

The Asset Summary Reporting (ASR) specification [[NISTIR-7848](#)] provides a format for capturing summary information about one or more assets. Specifically, it provides the ability to express a collection of records from some defined data source and map them to some set of assets. As a result, this specification may be useful for capturing the software installed on an endpoint, its relevant attributes, and associating it with a particular endpoint.

[11.5.1.1.2.](#) Software Identification Tags

SWID tag were previously introduced in [Section 11.1.3.1.2](#). As stated before, SWID tags are ideally deployed to an endpoint during software installation. In the less ideal case, they may also be generated based on information retrieved from a proprietary software installation data store. At minimum, SWID tag must contain an

identifier for each unit of installed software. Given this, SWID tags may be a viable way for SACM to express detailed information about the software installed on an endpoint.

[11.5.2.](#) Collected Platform Configuration Posture Attributes

Configurations associated with a software instance associated with an endpoint

A list of the configuration posture attributes associated with the actual values collected from the endpoint during the assessment as required/expressed by any related guidance. Additionally, each configuration posture attribute is associated with the installed software instance it pertains to.

[11.5.2.1.](#) Related Work

[11.5.2.1.1.](#) Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in [Section 11.2.1.1.2.1](#). As mentioned earlier, the OVAL System Characteristics platform extensions provide a catalog of the posture attributes that can be collected and assessed in the form of OVAL Items. These OVAL Items also serve as a model for representing posture attribute data and associated values that are collected off an endpoint. Furthermore, the OVAL System Characteristics model provides a system_info construct that captures information that identifies and characterizes the endpoint from which the posture attribute data was collected. Specifically, it includes operating system name, operating system version, endpoint architecture, hostname, network interfaces, and an extensible construct to support arbitrary additional information that may be useful in identifying the endpoint in an enterprise such as information capture in Asset Identification constructs. The OVAL System Characteristics model could serve as a useful starting point for representing posture attribute data collected from an endpoint although it may need to undergo some changes to satisfy the needs of the SACM community.

[11.5.2.1.2.](#) NETCONF-Based Collection

Introduced earlier in [Section 11.4.2.1.2](#), NETCONF defines a protocol for managing and retrieving posture attribute data from network infrastructure endpoints. NETCONF provides the <get-config> and <get> operations to retrieve the configuration data, and configuration data and state data respectively from a network infrastructure endpoint. Upon successful completion of these operations, the current posture attribute data of the network infrastructure endpoint will be made available. NETCONF also

provides a variety of filtering mechanisms (XPath, subtree, content matching, etc.) to trim down the posture attribute data that is collected from the endpoint. Given that NETCONF is widely adopted by network infrastructure vendors, it may be useful to consider this protocol as a standardized mechanism for collecting posture attribute data from network infrastructure endpoints.

As a side note, members of the OVAL Community have also developed a proposal to extend the OVAL Language to support the assessment of NETCONF configuration data [1]. The proposal leverages XPath to extract the posture attribute data of interest from the XML data returned by NETCONF. The collected posture attribute data can then be evaluated using OVAL Definitions and the results of the evaluation can be expressed as OVAL Results. While this proposal is not currently part of the OVAL Language, it may be worth considering.

[11.5.2.1.3](#). SNMP-Based Collection

The SNMP, previously introduced in [Section 11.4.2.1.3](#), defines a protocol for managing and retrieving posture attribute data from endpoints on a network [RFC3411]. SNMP provides three protocol operations [RFC3416] (GetRequest, GetNextRequest, and GetBulkRequest) for retrieving posture attribute data defined by MIB objects. Upon successful completion of these operations, the requested posture attribute data of the endpoint will be made available to the requesting application. Given that SNMP is widely adopted by vendors, and the MIBs that define posture attribute data on an endpoint are highly extensible, it may be useful to consider this protocol as a standardized mechanism for collecting posture attribute data from endpoints in an enterprise.

[11.6](#). Evaluation Guidance

[11.6.1](#). Configuration Evaluation Guidance

The evaluation guidance is applied by evaluators during posture assessment of an endpoint. This guidance must be able to reference or be associated with the following previously defined information elements:

- o configuration item identifiers,
- o platform configuration identifiers, and

- o collected Platform Configuration Posture Attributes.

[11.6.1.1.](#) Related Work

[11.6.1.1.1.](#) Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in [Section 11.2.1.1.2.1](#). The OVAL Definitions model provides an extensible framework for making assertions about the state of posture attribute data collected from an endpoint. Guidance written against this model consists of one or more OVAL Tests, that can be logically combined, where each OVAL Test defines what posture attributes should be collected from an endpoint (as OVAL Objects) and optionally defines the expected state of the posture attributes (as OVAL States). While the OVAL Definitions model may be a useful starting point for evaluation guidance, it will likely require some changes to decouple collection and evaluation concepts to satisfy the needs of the SACM community.

[11.6.1.1.2.](#) XCCDF Rule

A general description of XCCDF was provided in [Section 11.3.1.1.1](#). As noted there, an XCCDF document represents a checklist of items against which a given endpoint's state is compared and evaluated. An XCCDF Rule represents one assessed item in this checklist. A Rule contains both a prose description of the assessed item (either for presentation to the user in a tool's user interface, or for rendering into a prose checklist for human consumption) and can also contain instructions to support automated evaluation of the assessed item, if such automated assessment is possible. Automated assessment instructions can be provided either within the XCCDF Rule itself, or by providing a reference to instructions expressed in other languages, such as OVAL.

In order to support greater flexibility in XCCDF, checklists can be tailored to meet certain needs. One way to do this is to enable or disable certain rules that are appropriate or inappropriate to a given endpoint, respectively. For example, a single XCCDF checklist

might contain check items to evaluate the configuration of an endpoint's operating system. An endpoint deployed in an enterprise's DMZ might need to be locked down more than a common internal endpoint, due to the greater exposure to attack. In this case, some operating system configuration requirements for the DMZ endpoint might be unnecessary for the internal endpoint. Nonetheless, most configuration requirements would probably remain applicable to both environments (providing a common baseline for configuration of the given operating system) and thus be common to the checking instructions for both types of endpoints. XCCDF supports this by allowing a single checklist to be defined, but then tailored to the needs of the assessed endpoint. In the previous example, some Rules

that apply only to the DMZ endpoint would be disabled during the assessment of an internal endpoint and would not be exercised during the assessment or count towards the assessment results. To accomplish this, XCCDF uses the CPE Applicability Language. By enhancing this applicability language to support other aspects of endpoint characterization (see [Section 11.3.1.3](#)), XCCDF will also benefit from these enhancements.

In addition, XCCDF Rules also support parameterization, allowing customization of the expected value for a given check item. For example, the DMZ endpoint might require a password of at least 12 characters, while an internal endpoint may only need 8 or more characters in its password. By employing parameterization of the XCCDF Rule, the same Rule can be used when assessing either type of endpoint, and simply be provided with a different target parameter each time to reflect the different role-based requirements. Sets of customizations can be stored within the XCCDF document itself: XCCDF Values store parameters values that can be used in tailoring, while XCCDF Profiles store sets of tailoring instructions, including selection of certain Values as parameters and the enabling and disabling of certain Rules. The tailoring capabilities supported by XCCDF allow a single XCCDF document to encapsulate configuration evaluation guidance that applies to a broad range of endpoint roles.

[11.7.](#) Evaluation Result Reporting

[11.7.1.](#) Configuration Evaluation Results

The evaluation guidance applied during posture assessment of an

endpoint to customize the behavior of evaluators. Guidance can be used to define specific result output formats or to select the level-of-detail for the generated results. This guidance must be able to reference or be associated with the following previously defined information elements:

- o configuration item identifiers,
- o platform configuration identifiers, and
- o collected Platform Configuration Posture Attributes.

[11.7.1.1](#). Related Work

[11.7.1.1.1](#). XCCDF TestResults

A general description of the eXtensible Configuration Checklist Description Format (XCCDF) was provided in section [Section 11.3.1.1.1](#). The XCCDF TestResult structure captures the

outcome of assessing a single endpoint against the assessed items (i.e., XCCDF Rules) contained in an XCCDF instance document. XCCDF TestResults capture a number of important pieces of information about the assessment including:

- o The identity of the assessed endpoint. See [Section 11.3.1.1.2](#) for more about XCCDF structures used for endpoint identification.
- o Any tailoring of the checklist that might have been employed. See [Section 11.6.1.1.2](#) for more on how XCCDF supports tailoring.
- o The individual results of the assessment of each enabled XCCDF Rule in the checklist. See [Section 11.6.1.1.2](#) for more on XCCDF Rules.

The individual results for a given XCCDF Rule capture only whether the rule "passed", "failed", or experienced some exceptional condition, such as if an error was encountered during assessment. XCCDF 1.2 Rule results do not capture the actual state of the endpoint. For example, an XCCDF Rule result might indicate that an endpoint failed to pass requirement that passwords be of a length greater than or equal to 8, but it would not capture that the

endpoint was, in fact, only requiring passwords of 4 or more characters. It may, however, be possible for a user to discover this information via other means. For example, if the XCCDF Rule uses an OVAL Definition to effect the Rule's evaluation, then the actual endpoint state may be captured in the corresponding OVAL System Characteristics file.

The XCCDF TestResult structure does provide a useful structure for understanding the overall assessment that was conducted and the results thereof. The ability to quickly determine the Rules that are not complied with on a given endpoint allow administrators to quickly identify where remediation needs to occur.

[11.7.1.1.2.](#) Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in [Section 11.2.1.1.2.1](#). OVAL Results provides a model for expressing the results of the assessment of the actual state of the posture attribute values collected of an endpoint (represented as an OVAL System Characteristics document) against the expected posture attribute values (defined in an OVAL Definitions document). Using OVAL Directives, the granularity of OVAL Results can also be specified. The OVAL Results model may be useful in providing a format for capturing the results of an assessment.

[11.7.1.1.3.](#) Asset Summary Reporting

A general overview of ASR was provided previously in [Section 11.5.1.1.1](#). As ASR provides a way to report summary information about assets, it can be used within the SACM Architecture to provide a way to aggregate asset information for later use. It makes no assertions about the data formats used by the assessment, but rather provides an XML, record-based way to collect aggregated information about assets.

By using ASR to collect this summary information within the SACM Architecture, one can provide a way to encode the details used by various reporting requirements, including user-definable reports.

[11.7.1.1.4.](#) ARF

A general overview of ARF was provided previously in [Section 11.3.1.2.1](#). Because ARF is data model agnostic, it can provide a flexible format for exchanging collection and evaluation information from endpoints. It additionally provides a way to encode relationships between guidance and assets, and as such, can be used to associate assessment results with guidance. This could be the guidance that directly triggered the assessment, or for guidance that is run against collected posture attributes located in a central repository.

[11.7.2](#). Software Inventory Evaluation Results

The results of an evaluation of an endpoint's software inventory against an authorized software list. The authorized software list represents the policy for what software units are allowed, prohibited, and mandatory for an endpoint.

[12](#). Acknowledgements

Many of the specifications in this document have been developed in a public-private partnership with vendors and end-users. The hard work of the SCAP community is appreciated in advancing these efforts to their current level of adoption.

Over the course of developing the initial draft, Brant Cheikes, Matt Hansbury, Daniel Haynes, and Charles Schmidt have contributed text to many sections of this document.

[13](#). IANA Considerations

This memo includes no request to IANA.

[14](#). Security Considerations

Posture Assessments need to be performed in a safe and secure manner. In that regard, there are multiple aspects of security that apply to

the communications between components as well as the capabilities themselves. Due to time constraints, this information model only contains an initial listing of items that need to be considered with respect to security. This list is not exhaustive, and will need to be augmented as the model continues to be developed/refined.

Initial list of security considerations include:

Authentication: Every component and asset needs to be able to identify itself and verify the identity of other components and assets.

Confidentiality: Communications between components need to be protected from eavesdropping or unauthorized collection. Some communications between components and assets may need to be protected as well.

Integrity: The information exchanged between components needs to be protected from modification. some exchanges between assets and components will also have this requirement.

Restricted Access: Access to the information collected, evaluated, reported, and stored should only be viewable/consumable to authenticated and authorized entities.

[15.](#) References

[15.1.](#) Normative References

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

[15.2.](#) Informative References

[CCE] The National Institute of Standards and Technology, "Common Configuration Enumeration", 2014, <<http://nvd.nist.gov/CCE/>>.

Systems Agency, "Control Correlation Identifier", 2014, <<http://iase.disa.mil/cci/>>.

[CPE-WEBSITE]

The National Institute of Standards and Technology, "Common Platform Enumeration", 2014, <<http://scap.nist.gov/specifications/cpe/>>.

[CVE-WEBSITE]

The MITRE Corporation, "Common Vulnerabilities and Exposures", 2014, <<http://cve.mitre.org/about/>>.

[I-D.camwinget-sacm-requirements]

Cam-Winget, N., "Secure Automation and Continuous Monitoring (SACM) Requirements", [draft-camwinget-sacm-requirements-04](#) (work in progress), June 2014.

[I-D.ietf-sacm-terminology]

Waltermire, D., Montville, A., Harrington, D., and N. Cam-Winget, "Terminology for Security Assessment", [draft-ietf-sacm-terminology-04](#) (work in progress), May 2014.

[I-D.ietf-sacm-use-cases]

Waltermire, D. and D. Harrington, "Endpoint Security Posture Assessment - Enterprise Use Cases", [draft-ietf-sacm-use-cases-07](#) (work in progress), April 2014.

[IM-LIAISON-STATEMENT-NIST]

Montville, A., "Liaison Statement: Call for Contributions for the SACM Information Model to NIST", May 2014, <<http://datatracker.ietf.org/liaison/1329/>>.

[ISO.18180]

"Information technology -- Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2", ISO/IEC 18180, 2013, <http://standards.iso.org/ittf/PubliclyAvailableStandards/c061713_ISO_IEC_18180_2013.zip>.

[ISO.19770-2]

"Information technology -- Software asset management -- Part 2: Software identification tag", ISO/IEC 19770-2, 2009.

[NISTIR-7275]

Waltermire, D., Schmidt, C., Scarfone, K., and N. Ziring, "Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2", NISTIR 7275r4, March 2013, <http://csrc.nist.gov/publications/nistir/ir7275-rev4/nistir-7275r4_updated-march-2012_clean.pdf>.

[NISTIR-7693]

Wunder, J., Halbardier, A., and D. Waltermire, "Specification for Asset Identification 1.1", NISTIR 7693, June 2011, <<http://csrc.nist.gov/publications/nistir/ir7693/NISTIR-7693.pdf>>.

[NISTIR-7694]

Halbardier, A., Waltermire, D., and M. Johnson, "Specification for the Asset Reporting Format 1.1", NISTIR 7694, June 2011, <<http://csrc.nist.gov/publications/nistir/ir7694/NISTIR-7694.pdf>>.

[NISTIR-7695]

Cheikes, B., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Naming Specification Version 2.3", NISTIR 7695, August 2011, <<http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf>>.

[NISTIR-7696]

Parmelee, M., Booth, H., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Name Matching Specification Version 2.3", NISTIR 7696, August 2011, <<http://csrc.nist.gov/publications/nistir/ir7696/NISTIR-7696-CPE-Matching.pdf>>.

[NISTIR-7697]

Cichonski, P., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Dictionary Specification Version 2.3", NISTIR 7697, August 2011, <<http://csrc.nist.gov/publications/nistir/ir7697/NISTIR-7697-CPE-Dictionary.pdf>>.

[NISTIR-7698]

Waltermire, D., Cichonski, P., and K. Scarfone, "Common Platform Enumeration: Applicability Language Specification Version 2.3", NISTIR 7698, August 2011,

<<http://csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf>>.

[NISTIR-7848]

Davidson, M., Halbardier, A., and D. Waltermire, "Specification for the Asset Summary Reporting Format 1.0", NISTIR 7848, May 2012, <http://csrc.nist.gov/publications/drafts/nistir-7848/draft_nistir_7848.pdf>.

[OVAL-LANGUAGE]

Baker, J., Hansbury, M., and D. Haynes, "The OVAL Language Specification version 5.10.1", January 2012, <<https://oval.mitre.org/language/version5.10.1/>>.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, [RFC 3411](#), December 2002.

[RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, [RFC 3416](#), December 2002.

[RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, [RFC 3418](#), December 2002.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", [RFC 3444](#), January 2003.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", [RFC 4287](#), December 2005.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", [RFC 4949](#), August 2007.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.

- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011.

Waltermire & Watson

Expires January 4, 2015

[Page 68]

Internet-Draft Endpoint Assessment Information Model

July 2014

[SP800-117]

Quinn, S., Scarfone, K., and D. Waltermire, "Guide to Adopting and Using the Security Content Automation Protocol (SCAP) Version 1.2", SP 800-117, January 2012, <<http://csrc.nist.gov/publications/drafts/800-117-R1/Draft-SP800-117-r1.pdf>>.

[SP800-126]

Waltermire, D., Quinn, S., Scarfone, K., and A. Halbardier, "The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2", SP 800-126, September 2011, <<http://csrc.nist.gov/publications/nistpubs/800-126-rev2/SP800-126r2.pdf>>.

15.3. URIs

- [1] <https://github.com/OVALProject/Sandbox/blob/master/x-netconf-definitions-schema.xsd>

Authors' Addresses

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

Email: david.waltermire@nist.gov

Kim Watson

United States Department of Homeland Security
DHS/CS&C/FNR
245 Murray Ln. SW, Bldg 410
MS0613
Washington, DC 20528
USA

Email: kimberly.watson@hq.dhs.gov