Secure Automation and Continuous Monitoring (SACM) Architecture
draft-ietf-sacm-architecture-03

Abstract

This document defines a reference architecture for standardization of interfaces, protocols, and information models related to security automation and continuous monitoring. It describes the basic architecture, components, and their interfaces defined to enable the collection, acquisition, and verification of Posture and Posture Assessments.

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 September 10, 2015.

Copyright Notice

Copyright (c) 2015 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.
1. Introduction

Several data models and protocols are in use today that allow different applications to perform the collection, acquisition, and assessment of posture. These applications can vary from being focused on general system and security management to specialized configuration, compliance, and control systems. With an existing varied set of applications, there is a strong desire to standardize data models, protocols, and interfaces to better allow for the automation of such data processes.
This document addresses general and architectural requirements defined in [I-D.ietf-sacm-requirements]. This document describes an architecture to enable standardized collection, acquisition, and verification of Posture and Posture Assessments. This architecture includes the components and interfaces that can be used to better identify the Information Model and type(s) of transport protocols needed for communication.

This document uses terminology defined in [I-D.ietf-sacm-terminology].

2. Problem Statement

Securing information and the systems that store, process, and transmit that information is a challenging task for organizations of all sizes, and many security practitioners spend much of their time on manual processes. Administrators can’t get technology from disparate sources to work together; they need information to make decisions, but the information is not available. Everyone is collecting the same data, but storing it as different information. Administrators therefore need to collect data and craft their own information, which may not be accurate or interoperable because it’s customized by each administrator, not shared.

Security automation and continuous monitoring require a large and broad set of mission and business processes; to make the most effective use of technology, the same data must support multiple processes. The need for complex characterization and assessment necessitates components and functions that interoperate and can build off each other to enable far-ranging and/or deep-diving analysis.

3. Architectural Overview

At a high level, the architecture describes ‘How’ and ‘Where’ information and assessment of posture may be collected, processed, assessed, exchanged, and/or stored. Three main functional components are defined: a Posture Assessment Information Consumer (Cs), a Posture Assessment Information Provider (P), and a Controller (Cr) used to facilitate some of the security functions such as authentication and authorization and other metadata functions.
3.1. Component Roles

An endpoint, as defined in [I-D.ietf-sacm-terminology], can function in two primary ways: as the target of an assessment, and/or as a functional component of the SACM architecture that can instantiate one or more capabilities (see Section 4). In the SACM architecture,

Figure 1: Simple Architectural Model
individual endpoints may be a target endpoint, or a component, or both simultaneously. An endpoint acting as a component may perform one or more roles. Components can take on the role(s) of Posture Assessment Information Provider, Posture Assessment Information Consumer, and/or Controller.

3.1.1. Posture Assessment Information Provider

The Posture Assessment Information Provider (P or Provider) is the component that contributes Posture Assessment Information and/or Guidance either spontaneously or in response to a request. A Provider can be a Posture Evaluator, Posture Collector, Data Store (see Section 4.2), or an application that has aggregated Posture Assessment Information that can be shared.

The Provider implements the capabilities and functions that must be handled to share or provide Posture Assessment information.

A Provider may provide information spontaneously, or in response to a direct request from a Consumer. The information may be filtered or truncated to provide a subset of the requested information to honor the request. This truncation may be performed based on the Consumer’s request and/or the Provider’s ability to filter. The latter case may be due to security considerations (e.g. authorization restrictions due to domain segregation, privacy, etc.).

The Provider may only be able to share the Posture Assessment Information using a specific data model and protocol. It may use a standard data model and/or protocol, a non-standard data model and/or protocol, or any combination of standard and non-standard data models and protocols. It may also choose to advertise its capabilities through a metadata abstraction within the data model itself, or through the use of the registration function of the Controller (see Section 3.1.3).

The Provider must be authorized to provide the Posture Assessment Information and further, be authorized to do so with specific data models and protocols and/or for specific consumers.

3.1.2. Posture Assessment Information Consumer

The Posture Assessment Information Consumer (C or Consumer) is the component that requests or accepts Posture Assessment Information and/or Guidance. A Consumer can be a Posture Evaluator, Report Generator, Data Store (see Section 4.2), or an application that consumes Posture Assessment Information in order to perform another function.
As described in Section 2.2 of the SACM Use Cases [I-D.ietf-sacm-use-cases], several usage scenarios are posed with different application types requesting posture assessment information. Whether it is a configuration verification system; a checklist verification system; or a system for detecting posture deviations, compliance or vulnerabilities, they all need to acquire information about Posture Assessment. The architectural component performing such requests is a Consumer.

The Consumer implements the capabilities and functions that must be handled in order to facilitate a Posture Assessment Information Request. Requests can be either for a single posture attribute or a set of posture attributes; those attributes can be the raw information, or an evaluated or assessed state based upon that information. The Consumer may further choose to query for the information directly (one-time query), or to request for updates to be provided as the Posture Assessment Information changes (subscription). A request could be made directly to an explicitly identified Provider, but a Consumer may also desire to obtain the information without having to know the available Providers.

There may be instances where a Consumer may be requesting information from various Providers and, due to its policy or application requirements, may need to be better informed of the Providers and their capabilities. In those use cases, a Consumer may also request to discover the respective capabilities of those Providers using the discovery function of the Controller (see Section 3.1.3) or may request metadata reflecting the capabilities of the Providers.

The Controller (described below) must authorize a Consumer to acquire the information it is requesting. The Consumer may also be subject to limits or constraints on the numbers, types, sizes, and rate of requests.

3.1.3. Controller

The Controller (Cr or Controller) is a component defined to facilitate information sharing and to execute on security functions and overall SACM management and control system functions including:

Authentication: The authentication of Consumers and Providers independent of the actual information-sharing communication channel. This supports use cases where:

* Consumers may request information independent of knowing the identities of the Providers.
* Providers may want to share the information without prior solicitation.

The architecture must account for an abstraction where a Controller may be defined to effect the authentication of the Consumers and Providers independent of the actual information-sharing communication channel.

Authorization: The restriction of Posture Assessment Information sharing between the Consumers and Providers. At minimum, a management function must define the necessary policies.

Identity Management: Since Identity Management for authentication and authorization policies is best performed via a centralized component, the Controller also facilitates this function.

The Controller needs to be able to identify the endpoints participating as SACM components and the roles that they play. Similar to how access control may be effected via Authentication, Authorization, and Accounting Systems (e.g. AAA services), the same principle is defined; as AAA services depend on Identity Management services, the Controller will need a similar function and interface to Identity Management services.

Registration/Discovery: The discovery of what Providers are available, what information a Provider can share, and how it can be requested / communicated. A discovery mechanism is required to facilitate interaction with Providers that may have different Posture Assessment Information and potentially limited, or a rich set of, ways in which they can share the information.

Through the use of a discovery mechanism, Consumers can have visibility into the Providers present, the type(s) of Posture Assessment Information available, and how it can be requested. Similarly, a Provider may need to publish what Posture Assessment Information it can share and how it can share it (e.g. protocol, filtering capabilities, etc.). Enabling this function through a Controller or through metadata publication also allows for the distinct definition of security considerations (e.g. authorized registration / publication of capabilities by Providers) beyond how a Provider may define its own capability.

Beyond the control and management functions for the SACM system, a Controller may also provide proxy or broker or repository (and possibly routing) capabilities in the data plane (see Section 4.1). In the deployment scenario where Providers do not assert the need to know their Consumers and/or vice versa, the Controller can thus provide the appropriate functions to ensure the Posture Assessment
Information is appropriately communicated from the Providers to the authorized Consumers.

The Controller, acting as a management control plane, helps define how to manage an overall SACM system that allows for Consumers to obtain the desired Posture Assessment Information without the need to distinctly know and establish one (Consumer) to many (Provider) connections. Similarly, a Provider may not need to distinctly know and establish one (Provider) to many (Consumer) connections; e.g. the Controller enables the means to allow a SACM system to support many to many connections. Note that the Controller also allows for the direct discovery and connection between a Consumer and Provider.

As a SACM component, the Controller may be instantiated within a system or device acting as a Provider or a Consumer (or both), or as its own distinct Controller entity. In a rich SACM environment, it is feasible to instantiate a Controller that provides both the management (and control) functions for SACM as well as provide the proxying, brokering, and/or repository capabilities for the actual data, e.g. Posture Assessment Information flow. Note that Controllers may be implemented to only provide the management and control functions or only the data flow capabilities or both.

3.2. Interfaces between Consumers, Providers, and Controllers

As shown in Figure 1, communication can proceed with the following interfaces and expected functions and behaviors:

A: interface "A" shown in Figure 1 handles the management and control functions that are needed to establish, at minimum, a secure communication between Consumers and Providers. The interface must also handle the functions to allow for the discovery and registration of the Providers as well as the ways in which Posture Assessment Information can be provided (or requested).

B: interface "B" shown in Figure 1 enables Providers to share their Posture Assessment Information spontaneously; similarly, it enables Consumers to request information without having to know the identities (or reachability) of all the Providers that can fulfill Consumers’ requests.

C: interface "C" shown in Figure 1 illustrates the ability and desire for Consumers and Providers to be able to communicate directly when a Provider is sharing Posture Assessment Information directly to a Consumer. The interface allows for the different data models and protocols to be used between a Consumer and a Provider with the expectation that the appropriate authentication and authorization mechanisms have been employed to establish a secure
communication link between the Consumer and the Provider. Typically, it is expected that the secure link establishment occurs as a management or control function through the abstracted Controller role (e.g. the Controller could be a broker or could be embedded in a Consumer or a Provider).

A variety of protocols, such as SNMP, NETCONF, NEA protocols [RFC5209], and other similar interfaces, may be used for collection of data from the target endpoints by the Posture Information Provider. Those interfaces are outside the scope of SACM.

4. Component Capabilities

SACM components offer a variety of capabilities which may be instantiated on a single endpoint or on separate standalone endpoints providing various roles.

4.1. Control Plane Capabilities

Control plane capabilities represent various services offered by the Controller to the Providers and Consumers to facilitate sharing of information. A Controller may have Broker, Proxy, or Repository capabilities, or any combination thereof.

Broker: Intermediary negotiating connection between Provider and Consumer. A Controller acting as a Broker:

* Receives a request for information from a Consumer and instructs the Consumer where and how to retrieve the requested information.

* Receives a publication request from a Provider and instructs the Provider where and how to deliver the published information.

The information itself is neither distributed nor stored by the Controller.

Proxy: Intermediary negotiating on behalf of a Consumer or Provider. A Controller acting as a Proxy:

* Receives a request for information from a Consumer, retrieves the information from the appropriate Providers, and provides the information to the Consumer.

* Receives a publication request from a Provider, accepts the published information, and distributes it to appropriate consumers.
The information itself is distributed by, but not stored by, the Controller.

Repository: Intermediary receiving and storing data from a Provider, and providing stored data to a Consumer. A Controller acting as a Repository:

* Receives a request for information from a Consumer, retrieves the information from its data stores, and provides the information to the Consumer.

* Receives a publication request from a provider, stores the published information, and distributes it to appropriate Consumers.

The information itself is both handled by and stored by the Controller.

4.2. Data Plane Capabilities

Data plane capabilities represent the ability of a Provider or Consumer to perform a SACM-related task. For example, the Collector capability indicates that a Provider can perform Collection tasks; the Evaluator capability indicates that a Consumer can perform Evaluation tasks.

4.2.1. Collector

A collector consumes Guidance and/or other Posture Assessment Information; it provides Posture Assessment Information. Collectors may be internal or external.

4.2.1.1. Internal Collector

An internal collector is a collector that runs on the endpoint and collects posture information locally.

4.2.1.2. External Collector

An external collector is a collector that observes endpoints from outside. These collectors may be configured and operated to manage assets for reasons including, but not limited to, posture assessment. Collectors that are not primarily intended to support posture assessment (e.g. intrusion detection systems) may still provide information that speaks to endpoint posture (e.g. behavioral information).

Examples:
- A RADIUS server, which collects information about which endpoints have logged onto the network
- A network profiling system, which collects information by discovering and classifying network nodes
- A Network Intrusion Detection System (NIDS) sensor, which collects information about endpoint behavior by observing network traffic
- A vulnerability scanner, which collects information about endpoint configuration by scanning endpoints
- A hypervisor, which collects information about endpoints running as virtual guests in its host environment
- A management system that configures and installs software on the endpoint, which collects information based on its provisioning activities

4.2.1.3. Collector Interactions With Target Endpoints

TODO - examples of endpoint interactions with local internal collector (e.g. NEA client), endpoint with remote internal collector (SNMP query), and external collector (sensor)

4.2.2. Evaluator

An evaluator consumes Posture Assessment Information, Evaluation Results, and/or Guidance; it provides Evaluation Results. An evaluator may consume endpoint attribute assertions, previous evaluations of posture attributes, or previous reports of Evaluation Results.

TODO: update the terminology doc to reflect this definition

Example: a NEA posture validator [RFC5209]

[jmf- a NEA posture validator is not an example of this definition. A NEA posture assessment is, maybe?]

[cek-Why isn’t a NEA posture validator an example?]

4.2.3. Report Generator

A report generator consumes Posture Assessment Information, Evaluation Results, and/or Guidance; it provides reports. These reports are based on:
Endpoint Attribute Assertions, including Evaluation Results

Other Reports (e.g., a weekly report may be created from daily reports)

It may summarize data continually, as the data arrives. It also may summarize data in response to an ad hoc query.

4.2.4. Data Store

A data store consumes any data; it provides any data.

5. Example Illustration of Capabilities and Workflow

TODO: once the group reaches consensus on content for the previous sections, revise all this text based upon the agreed-upon architecture

SACM’s focus is on the automation of collection, verification and update of system security configurations pertaining to endpoint assessment. In order to carry out these tasks, the architectural components shown in Figure 1 can be further refined as:

Posture Assessment Information Providers: a Provider may be dedicated to perform either the collection, aggregation or evaluation of one or more posture attributes whose results can be
conveyed to a Posture Assessment Information Consumer. In this example form of the SACM architecture model, these are shown as Collection, Evaluation, and Results Providers. Note that there may be posture attributes or posture assessment information that articulates Guidance information which may or may not be present in the architecture.

Posture Assessment Information Consumers: a Consumer may request or receive one or more posture attributes or posture assessment information from a Posture Assessment Information Provider for their own use. In this example form of the SACM architecture model, these are shown as Collection, Evaluation, and Results Consumers. Note that there may be posture attributes or posture assessment information articulating Guidance information which may or may not be present in the architecture to be provided or consumed.

Data Stores: a Data Store is both a Provider and a Consumer, storing one or more posture attributes or assessments for endpoints. It should be understood that these repositories interface directly to a Provider or Consumer (and Guidance) but the interfaces used to interact between them is outside the scope of SACM (e.g. no interface arrows are shown in the architecture).

Figure 3 illustrates an example flow for how Posture Assessment Information may flow.
Figure 3: Example Posture Information Flow
6. Acknowledgements

The authors would like to thank Jim Bieda, Henk Birkholz, Jessica Fitzgerald-McKay, Trevor Freeman, Adam Montville, and David Waltermire for participating in architecture design discussions, reviewing, and contributing to this draft.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

The SACM architecture defines three main components that interface with each other both for management and control (in the control plane) and for the sharing of Posture Assessment Information. Considerations for transitivity of trust between a Provider and Consumer can be made if there is a well understood trust between the Provider and the Controller and between the Consumer and Controller. The trust must include strong mutual authentication, at minimum, between the Provider and Controller and between the Consumer and Controller.

To address potential Man-in-the-Middle (MitM) attacks, it is also strongly recommended that the communications be secured to include replay protection and message integrity (e.g. transport integrity and if required, data integrity). Similarly, to avoid potential message tampering, confidentiality should also be provided.

As the Controller provides the security functions for the SACM system, the Controller should provide strong authorizations based on either or both business and regulatory policies to ensure that only authorized Consumers and obtaining Posture Assessment Information from authorized Providers. It is presumed that once authenticated and authorized, the Provider, Controller or Consumer is deemed trustworthy; though note that it is possible that the modules or devices hosting the SACM components may be compromised as well (e.g. due to malware or tampering); however, addressing that level of trustworthiness is out of scope for SACM.

As the data models defined through the interfaces are transport agnostic, the Posture Assessment Information data in the interfaces may leverage the transport security properties as the interfaces are transported between the Provider, Consumer and Controller. However, there may be other devices, modules or components in the path between
the Provider, Consumer and Controller that may observe the interfaces flowing through them.

9. References

9.1. Normative References

[I-D.ietf-sacm-requirements]

[I-D.ietf-sacm-terminology]

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


9.2. Informative References


Authors' Addresses

Nancy Cam-Winget (editor)
Cisco Systems
3550 Cisco Way
San Jose, CA 95134
US

Email: ncamwing@cisco.com
Lisa Lorenzin  
Pulse Secure  
2700 Zanker Rd, Suite 200  
San Jose, CA 95134  
US  
Email: llorenzin@pulsesecure.net

Ira E McDonald  
High North Inc  
PO Box 221  
Grand Marais, MI 49839  
US  
Email: blueroofmusic@gmail.com

Aaron Woland  
Cisco Systems  
1900 South Blvd. Suite 200  
Charlotte, NC 28203  
US  
Email: loxx@cisco.com
SACM Information Model
draft-ietf-sacm-information-model-01

Abstract

This document proposes an information model for SACM.

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 August 24, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

1. Introduction ........................................ 4
   1.1. Changes in Revision 01 .......................... 5

2. Problem Statement ................................... 6
   2.1. Mapping to SACM Use Cases ...................... 7
   2.2. Referring to an Endpoint ....................... 8
   2.3. Dealing with Uncertainty ....................... 8

3. Conventions used in this document .................. 9
   3.1. Requirements Language .......................... 9

4. Elements of the SACM Information Model ............. 9
   4.1. Software Component ............................ 12
   4.2. Software Instance .............................. 13
   4.3. Hardware Component ............................. 14
   4.4. Hardware Instance .............................. 14
   4.5. Network Interface .............................. 14
   4.6. Address ........................................ 15
   4.7. Identity ....................................... 16
   4.8. Location ....................................... 16
   4.9. Endpoint ....................................... 17
   4.10. Endpoint Attribute Assertion ................. 17
   4.10.1. Form and Precise Meaning .................. 17
   4.11. Attribute-Value Pair ......................... 18
   4.11.1. Posture Attribute .......................... 19
   4.11.2. Identifying Attributes ..................... 19
   4.11.3. Evaluation Result ........................... 21
   4.12. Report ........................................ 21
   4.13. SACM Component ................................ 22
   4.13.1. External Attribute Collector ............... 22
   4.13.2. Evaluator .................................. 22
   4.13.3. Report Generator ........................... 23
   4.15. Guidance ...................................... 23
   4.15.1. Internal Collection Guidance ............... 24
   4.15.2. External Collection Guidance ............... 24
   4.15.3. Evaluation Guidance ......................... 24
   4.15.4. Retention Guidance ......................... 24
   4.15.5. Reporting Guidance ......................... 24
   4.16. Provenance of Information ..................... 24
   4.17. Endpoint ....................................... 24
   4.17.1. Endpoint Identity .......................... 25
   4.17.2. Software Component ........................ 25
   4.17.2.1. Unique Software Identifier ............... 26
   4.18. User .......................................... 26
   4.18.1. User Identity ................................ 26

5. SACM Usage Scenario Example ....................... 26
   5.1. Graph Model for Detection of Posture Deviation .. 27
   5.1.1. Components .................................. 27
5.1.2. Identifiers ................................................. 27
5.1.3. Metadata .................................................. 28
5.1.4. Relationships between Identifiers and Metadata ....... 29
5.2. Workflow ..................................................... 29
6. Acknowledgements ............................................. 30
6.1. Contributors ............................................... 30
7. IANA Considerations ........................................... 30
8. Security Considerations ...................................... 30
9. References ..................................................... 31
  9.1. Normative References .................................... 31
  9.2. Informative References .................................. 31
Appendix A. Security Automation with TNC IF-MAP .............. 37
  A.1. What is Trusted Network Connect? ...................... 37
  A.2. What is TNC IF-MAP? ..................................... 37
  A.3. What is the TNC Information Model? .................... 38
Appendix B. Text for Possible Inclusion in the Terminology Draft 39
  B.1. Terms and Definitions ................................... 39
  B.1.1. Pre-defined and Modified Terms ...................... 39
  B.1.2. New Terms ............................................. 39
Appendix C. Text for Possible Inclusion in the Architecture or
  Use Cases ....................................................... 40
  C.1. Introduction .............................................. 40
  C.2. Core Principles ......................................... 41
  C.3. Architecture Assumptions ................................ 41
Appendix D. Text for Possible Inclusion in the Requirements
  Draft .......................................................... 45
  D.1. Problem Statement ....................................... 45
  D.2. Problem Scope ........................................... 45
Appendix E. Text With No Clear Home Yet ...................... 46
  E.1. Operations ................................................. 46
    E.1.1. Generalized Workflow ................................ 46
  E.2. From Information Needs to Information Elements ........ 47
  E.3. Information Model Elements ............................ 47
    E.3.1. Asset Identifiers .................................... 49
      E.3.1.2. Endpoint Identification .......................... 51
      E.3.1.3. Software Identification ........................... 52
      E.3.1.4. Hardware Identification ........................... 55
    E.3.2. Other Identifiers .................................... 55
      E.3.2.1. Platform Configuration Item Identifier .......... 55
      E.3.2.2. Configuration Item Identifier .................... 61
      E.3.2.3. Vulnerability Identifier ........................... 63
    E.3.3. Endpoint characterization ............................ 63
    E.3.4. Posture Attribute Expression ........................ 67
      E.3.4.2. Platform Configuration Attributes ............... 67
    E.3.5. Actual Value Representation ........................ 69
      E.3.5.1. Software Inventory ................................. 69
      E.3.5.2. Collected Platform Configuration Posture
                Attributes ............................................. 70
1. Introduction

This document defines a notional information model for endpoint posture assessment. It describes the information needed to perform certain assessment activities. The scope of the information model is to describe the structure of the information carried to realize the assessment. It is meant to be a basis for the development of specific data models. The terms information model and data model loosely align with the definitions 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:
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.

The SACM information model offers a loose coupling between providers and consumers of security information. A provider can relay what it observes or infers, without knowing which consumers will use the information, or how they will use it. A consumer need not know exactly which provider generated a piece of information, or by what method.

At the same time, a consumer *can* know these things, if necessary.

As things evolve, a provider can relay supplemental information. Some consumers will understand and benefit from the supplemental information; other consumers will not understand and will disregard it.

1.1. Changes in Revision 01

Added some proposed normative text.

For provenance:

- Added a class "Method"
- Added the produced-using relationship between an AVP and a method
- Added the produced-by relationship between a Guidance and a SACM Component
- Added the hosted-by relationship between a SACM Component and an Endpoint

asserted-by and summarized-by have been renamed to produced-by.

"User" is now "Account". If a user has different credentials, SACM cannot know that they belong to the same user. But, per Kim W, many organizations do have accounts that associate credentials.
The multiplicity of the based-on relationships has been corrected.

More relationships now have labels, per UML convention.

The diagram no longer has causal arrows. They had become redundant and were nonstandard and clutter.

Renamed "credential" to "identity", following industry usage. A credential includes proof, such as a key or password. A username or a distinguished name is called an "identity".

Removed Session, because an endpoint’s network activity is not SACM’s initial focus.

Removed Authorization, for the same reason

Added many-to-many relationship between Hardware Component and Endpoint, for clarity.

Added many-to-many relationship between Software Component and Endpoint, for clarity.

Added "contains" relationship between Network Interface and Network Interface.

Removed relationship between Network Interface and Account. The endpoint knows the identity it used to gain network access. The PDP also knows that. But they probably do not know the account.

Added relationship between Network Interface and Identity. The endpoint and the PDP will typically know the identity.

Made identity-to-account a many-to-one relationship.

2. Problem Statement

TODO: revise

(SACM requires 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 document 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. [What does "narrowed down the scope of this model" mean? - LL] )

(Wandw)
Administrators can’t get technology from disparate sources to work together; they need information to make decisions, but the information is not available. Everyone is collecting the same data, but storing it as different information. Administrators therefore need to collect data and craft their own information, which may not be accurate or interoperable because it’s customized by each administrator, not shared. A standard information model enables flexibility in collecting, storing, and sharing information despite platform differences.

A way is needed to exchange information that (a) has breadth, meaning the pieces of the notation are useful about a variety of endpoints and software components, and (b) has longevity, meaning that the pieces of the notation will stay useful over time.

When creating standards, it’s not sufficient to go from requirements directly to protocol; the standards must eliminate ambiguity in the information transported. This is the purpose of information models generally. The SACM problem space is about integrating many information sources. This information model addresses the need to integrate security components, support multiple data models, and provide interoperability in a way that is platform agnostic, scales, and works over time.

2.1. Mapping to SACM Use Cases

TODO: revise

(wandw)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 [removed] thru [removed] 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.ietf-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.(/wandw)
2.2. Referring to an Endpoint

How to refer to an endpoint is problematic. Ideally, an endpoint would have a unique identifier. These identifiers would have a one-to-one relationship with endpoints. Every observation of an endpoint, or inference about an endpoint would be labeled with its identifier.

However:

- An external posture attribute collector typically cannot observe the unique identifier directly. An external posture attribute collector should be able to report exactly what it has observed, unembellished. It should not have to "infer" which endpoint it has observed; that inference should be leavable to other SACM components. So, SACM cannot require that every observation include the unique endpoint identifier.

- Internal posture attribute collectors are not present on all endpoints. They are not present on "dumb" devices such as Internet of Things (IoT) devices, or on Bring Your Own Device (BYOD) devices. In these cases, *no* observers have direct access to the unique endpoint identifier.

- An endpoint identifier is generally subject to cloning, when a system image is cloned. Then it is no longer unique.

- Suppose the endpoint identifier is highly clone resistant -- such a unique certificate within a trusted platform module TPM). Even so, it is possible to replace all of the software -- for example, changing a Windows machine to a Linux machine. Is it still the same endpoint? For SACM purposes, it isn’t really the same endpoint.

So SACM components must be able to put disparate observations together and form a picture of an endpoint -- somewhat like a detective. The SACM information model must facilitate this.

2.3. Dealing with Uncertainty

With many information models, the information is considered certain. So it is OK to blur the difference between the representation and the thing represented.

In SACM, information is not certain. Attackers may develop countermeasures to fool some SACM components. Attackers may compromise some SACM components.
So the model must let SACM components and humans reason with uncertainty. There are no facts, only assertions.

SACM components must be able to cross check observations and inferences against each other. They should be able to give weight if an observation or inference is corroborated by more than one method. Although SACM will probably not prescribe *how* to do this cross checking, SACM should provide the components with provenance information.

SACM components must be able to consider the reputation of the observer or inferrer. That reputation should account for the method of observing or inferring, the implementer of the SACM component that made the observation or inference, and the compliance status of the endpoint on which the observation or inference was made. For example, if some observers are found to be vulnerable to a Day 1 exploit, observations from those observers deserve less weight. The details of reputation technology may be out of scope for SACM. However, again, SACM should provide components with provenance information.

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. Elements of the SACM Information Model

The SACM Information Model contains several elements of the architecture, including:

- SACM Components, which may be Collectors, Evaluators, etc. Collectors may be internal (performed within the endpoint itself) or external (performed outside of the endpoint, such as by a hypervisor or remote sensor)

- Guidance, which tells SACM components what to do

- Posture, in the form of posture attributes and evaluation results

- Additional information about the endpoint, such as a representation of a software component, endpoint identity, user identity, address, location, and authorization constraining the endpoint
The SACM Information Model does not (in this draft) specify how long information is retained. Historical information is modeled the same way as current information. Historical information may be represented differently in an implementation, but that difference would be in data models, not in the information model.

Figure 1 the the information model.
Figure 1: The Information Model

..... Above this line is the monitorable world
Note: UML 2 is specified by [UML].

TODO: update text to match new figure:

Need to be clear in the description that - of some of the relationships, will need some language and guidance to the interfaces and relationships we expect to have happen, MUSTs and SHOULDs, as well as explaining the extensibility that other relationships can exist, show examples of how that can happen. Others that we haven’t thought of yet, might be added by another RFC or in another way.

CEK: I suddenly wonder whether all of the relationships in the upper right corner of the diagram are needed. At present, AVPs mostly mention instances of the classes in the upper half. The only relationship an endpoint attribute assertion expresses is that a set of AVPs are all true of some endpoint. We don’t have a way to say that an address is bound to a particular interface. Such structures *can* be modeled, using YANG, for example. But do we require that? If we do, why? I do think we need to be able to relate a software instance to the software component, and a hardware instance to the hardware component.

The following subsections discuss the elements and relationships found in Figure 1.

4.1. Software Component

An endpoint contains and runs software components.

Relationship:

- If an endpoint has an instance of a software component, we say that the software component is "in" the endpoint. This is a shorthand.

Some software components are assets. "Asset" is defined in RFC4949 [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."

An examination of software needs to consider both (a) software assets and (b) software that may do harm. A posture attribute collector may not know (a) from (b). It is useful to define Software Component as the union of (a) and (b).

Examples of Software Assets:
o An application
o A patch
o The operating system kernel
o A boot loader
o Firmware that controls a disk drive
o A piece of JavaScript found in a web page the user visits

Examples of harmful software components:

o A malicious entertainment app
o A malicious executable
o A web page that contains malicious JavaScript
o A business application that shipped with a virus

Software components SHOULD be disjoint from each other. In other words, software components SHOULD be so defined that a given byte of software on an endpoint belongs to only one software component.

Different versions of the same piece of software MUST be modeled as different components. Software versioning is not built into the information model.

Each separately installable piece of software SHOULD be modeled as a component. Sometimes it may be better to divide more finely: what an installer installs MAY be modeled as several components.

A data model MAY identify a software component by parts of an ISO SWID tag.

4.2. Software Instance

Each copy of a piece of software is called a software instance. The configuration of a software instance is regarded as part of the software instance. Configuration can strongly affect security posture.

A data model MUST support the following relationships:

o A software instance is an "instance of" a software component.
A software instance is "in" an endpoint.

A data model MAY use ISO SWID tags to identify software instances.

4.3. Hardware Component

Hardware components may also be assets and/or harmful. For example, a USB port on a system may be disabled to prevent information flow into or out of a particular system; this provides an additional layer of protection that can complement software based protections. Other such assets may include access to or modification of storage media, hardware key stores, microphones and cameras. Like software assets, we can consider these hardware components both from the perspective of (a) an asset that needs protection and (b) an asset that can be compromised in some way to do harm.

A data model MAY designate a hardware component by its manufacturer and a part number.

4.4. Hardware Instance

A hardware instance is just an instance of a particular component.

A data model MUST support the following relationships:

- A hardware instance is an "instance of" a hardware component.
- A hardware instance is "in" an endpoint.

Hardware instances may need to be modeled because (a) an endpoint may have multiple instances of a hardware component, (b) a hardware instance may be compromised, whereas other instances may remain intact.

A data model MAY designate a hardware instance by its component and a unique serial number.

4.5. Network Interface

CEK: I am not sure how to use network interfaces for endpoint identification. As for compliance, is this too advanced for SACM at this time?

An endpoint generally has at least one network interface.

Interfaces nest. A virtual interface can nest in a physical interface.
A data model MUST support the following relationships:

- A network interface is "in" an endpoint.
- A network interface is "in" another network interface; this is for a nested interface. CEK: And this allows representing compliance policies that are worthwhile. But is this too advanced for the initial set of SACM RFCs?
- A network interface "acts for" an identity. This occurs, for example, when the network interface is online because of successful 802.1X. An internal collector may be aware of the identity. An external collector (such as a RADIUS server) will be aware of the identity.

4.6. Address

As used in this document, an address is any of:

- A layer 2 address; a data model MUST support MAC addresses, and MAY support others
- A layer 3 address; a data model MUST support IPv4 and IPv6 addresses, and MAY support others
- A layer 4 address; a data model MUST support an IP-address-protocol-port combination, where protocol is TCP or UDP. It MAY support others

Addresses from other layers may be added in the future.

These addresses are not necessarily globally unique. Therefore, a data model SHOULD allow an address to be qualified with a scope.

- A data model SHOULD allow qualifying a MAC address with its layer-2 broadcast domain. This MAY take the form of a VLAN ID and an administratively assigned string denoting the LAN.
- A data model SHOULD allow qualifying an IP address with an administratively assigned string denoting the IP routing domain.

A data model MUST support the following relationships:

- An address is "bound to" a network interface.
- An address is considered "bound to" an endpoint just if the address is "bound to" an interface that is "in" the endpoint.
4.7. Identity

An identity is the non-secret part of a credential. Examples are a username, an X.500 distinguished name, and a public key. Passwords, private keys, and other secrets are not considered part of an identity.

A data model MUST support the following relationships:

- An endpoint may "act for" an identity. This SHALL mean that the endpoint claims or proves that it has this identity. For example, if the endpoint is part of an Active Directory domain and Alice logs into the endpoint with her AD username (alice) and password, the endpoint "acts for" alice. An endpoint MAY "act for" more than one identity at once, such as a machine identity and a user identity.

- A identity may "belong to" an account. For example, an enterprise may have a database that maps identities to accounts. CEK: Is this relevant? I don’t see how we’d use the notion of an account in identifying an endpoint or in specifying compliance measurements to be taken.

4.8. Location

Location can be logical or physical. Location can be a clue to an endpoint’s identity.

A data model MUST support the following relationships:

- An endpoint may be "in" a location

- A location may be "in" one or more locations

- A network address may be "in" a location

- An account may be "in" a location; this would happen if the account represents a user, and a physical access control system reports on the user’s location

Examples of location:

- The switch, access point, VPN gateway, or cell tower to which the endpoint is linked

- The switch port where the endpoint is plugged in
The location of the endpoint’s IP address in the network topology

The geographic location of the endpoint (which is often self-reported)

A user location (may be reported by a physical access control system)

CEK: The last three examples seem too advanced for the first set of SACM RFCs. I do not know a notation that would be interoperable and useful for endpoint identification. Should we drop them?

CEK: If we do drop them, all we have left is the device and port at which the endpoint is linked to the network. Maybe we should regard that as a kind of address.

A data model MUST support switch + port number, access point, and VPN gateway as locations. The other examples are optional.

More than one of kind of location may pertain to an endpoint. Endpoint has a many-to-many relationship with Location.

4.9. Endpoint

An endpoint is the hollow center of the model. An endpoint is an abstract ideal. Any endpoint attribute assertion that mentions an endpoint mentions it by specifying identifying attributes. Even if there is one preferred endpoint identity, that is modeled as an identity. We do not anticipate any AVP whose attribute type is "endpoint".

4.10. Endpoint Attribute Assertion

Represents a direct observation by an attribute collector, or a conclusion drawn by an evaluator. It asserts that specified posture attribute-value pairs were true of an identified endpoint, at a specific time or throughout a specified time interval.

CEK: I think it asserts that there exists an endpoint for which all of the AVPs are true. The times and time intervals are features of the AVPs, not of the endpoint attribute assertion.

4.10.1. Form and Precise Meaning

An endpoint attribute assertion MUST have:

- One or more attribute-value pairs (AVPs) (see section XXX)
4.11. Attribute-Value Pair

DW: Pair is not a good word for this information since more than two data points are required. We should consider a better term.

An attribute-value pair (AVP) is a tuple of information asserting an observed aspect of endpoint posture and/or identity.

Each AVP MUST include the following data:

- Identification of a posture or identification attribute (see section [Posture Attribute])

- The value(s) associated with the posture or identification attribute at the time the observation was made.
  
  * The value MAY be structured. For example, it may something like XML.
  
  * Some attributes will be multi-valued. Data models implementing this information model MUST support multiple values.

- Temporal information bounding the observation. Temporal information MUST consist of one of the following:
  
  * A timestamp indicating the point-in-time the observation was made, or
  
  * An interval establishing the duration for which the value(s) have existed.
  
  * Information about the method used to derive the assertion. (see section [method])

If posture assertions are generated by monitoring a system for changes, describing the interval for which a given state was found to be consistent enables additional information to be expressed over a point-in-time observation. Since values may drift over time, expressing this additional information provides important context to downstream consumers of a posture assertion about the rate and time of change.
4.11.1. Posture Attribute

Posture Attribute is defined in [RFC5209] as "attribute describing the configuration or status (posture) of a feature of the endpoint. A Posture Attribute represents a single property of an observed state."

The set of attribute types MUST be extensible by other IETF standards, by other standards groups, and by vendors. How to express attribute types is not defined here, but is left to data models to address.

Example posture attributes: [ This needs to be updated ]

- TBD

4.11.2. Identifying Attributes

An identifying attribute is observation made at a specific point in time that can be used standalone or in combination with other observations to identify an endpoint. While all endpoint posture attributes can be used to establish an endpoint’s identity, not all attributes are equally suited. Primary identifying attributes are a subset of posture attributes that have the following desirable characteristics for use in identifying an endpoint:

- Available - Present on most platforms allowing for a breadth of interoperability between platforms.
- Stable - Change infrequently (e.g., assist with identifying an endpoint across network sessions)
- Authentic - High level of assurance, attributes can be authenticated

Other posture attributes that do not possess these characteristics are considered secondary identifying attributes. They can often be used to check an identity assertion or to further define the endpoint’s identity (e.g. configuration, software inventory, capability). Examples of primary identifying attributes:

- Certificates (e.g., device, user, drive, other hardware) MUST be provided if available on device; favored by consumers of the assertion
  - Others if available as associated with the session
ISSUE: Use whole cert or some aspect (e.g., public key)

- Hostname(s) - configured on the device; may be multiple and possibly of different kinds
  
  Consider how to specify type: multiple attributes for each type or type qualifier

- FQDN(s) (if available); some hosts may have multiple; also list DNS aliases
  
  Sourced from the DNS infrastructure
  
  ISSUE: Think about the security considerations of insecure DNS deployments

- Network Interface(s) (comprehensive, if possible)
  
  MAC and IP address(es)
  
  Including additional IPs assigned to the interface
  
  Interfaces may be nested (e.g., overlay networks - VLANs)
  
  NOTE: Need to verify that standardized methods are available to do this.

- Tool-specific Identifier - provided by the software making the assertion
  
  SHOULD be provided if available
  
  May be session scoped if the tool is not able to distinguish an endpoint across sessions
  
  May be omitted if the tool is unable to distinguish the endpoint across multiple assertions

- Identification information from bios/firmware (e.g., serial numbers, asset tags, VM id) that can be queried on the device using software. NOTE: Need to discuss more.

  Assets are compositional. Some components may have specific identifiers. It would be useful to indicate which component has a specific identifying attribute.
  
  Information should be made available by all manufacturers
If available should be provided as an attribute

- Other cryptographic information

4.11.3. Evaluation Result

Evaluation Results (see [I-D.ietf-sacm-terminology]) are modeled as Endpoint Attribute Assertions.

An Evaluation Result derives from one or more other Endpoint Attribute Assertions.

An example is: a NEA access recommendation [RFC5793]

An evaluator may be able to evaluate better if history is available. This is a use case for retaining Endpoint Attribute Assertions for a time.

An Evaluation Result may be retained longer than the Endpoint Attribute Assertions from which it derives. (Figure 1 does not show this.) In the limiting case, Endpoint Attribute Assertions are not retained. When as an Endpoint Attribute Assertion arrives, an evaluator produces an Evaluation Result. These mechanics are out of the scope of the Information Model.

4.12. Report

An Endpoint Attribute Assertion concerns a single endpoint. Assertions about a set of endpoints are also needed -- for example, for trend analysis and for reports read by humans. These assertions are termed "reports". SACM components will consume Endpoint Attribute Assertions and generate reports.

A report contains its provenance, with the same form and meaning as the provenance of an Endpoint Attribute Assertion.

A Report summarizes:

- Endpoint Attribute Assertions, which may include Evaluation Results
- Other Reports

A Report may routine or ad hoc.

Some reports may be machine readable. Machine readable reports may be consumable by SACM components and by automatic response systems (not specified by SACM).
4.13. SACM Component

Although SACM components are mainly covered by the SACM architecture, we have some remarks. TODO: Move them?

4.13.1. External Attribute Collector

An external collector is a collector that observes endpoints from outside. [kkw—many of these [collectors] are actually configured and operated to manage assets for reasons other than posture assessments. It is critical to bring them into this, so I like it...but does it matter if the [collector] isn’t intended to support posture assessment, but happens to have information that can be used by posture assessment collection consumers? Do we lump them together with collectors that are intended to support posture assessment but run external to the endpoint?] [jmf: ditto. The examples below are of things that would perform external collection].

[cek—to kkw’s comment, I think the purpose here is to capture their contribution to continuous monitoring. I don’t see the need to separate things whose primary job is monitoring from things whose primary job is something else. Is there a need?]

[cek—to jmf’s comment, that is what they are examples of; is a text change needed?]

Examples:

- A RADIUS server whereby an endpoint has logged onto the network
- A network profiling system, which discovers and classifies network nodes
- A Network Intrusion Detection System (NIDS) sensor
- A vulnerability scanner
- A hypervisor that peeks into the endpoint, the endpoint being a virtual machine
- A management system that configures and installs software on the endpoint

4.13.2. Evaluator

An evaluator can consume endpoint attribute assertions, previous evaluations of posture attributes, or previous reports of evaluation results. [kkw—I don’t think this conflicts with the definition in the
terminology doc re: that evaluation tasks evaluate posture attributes.]

[cek-I like the change. I think it *does* require a change in the terminology doc, though.]

Example: a NEA posture validator [RFC5209]

[jmf- a NEA posture validator is not an example of this definition. A NEA posture assessment is, maybe?]

[cek-Why isn’t a NEA posture validator an example?]

4.13.3. Report Generator

A report generator makes reports based on:

- Endpoint Attribute Assertions, including Evaluation Results
- Other Reports (a weekly report may be created from daily reports)

It may summarize data continually, as the data arrives. It also may summarize data in response to an ad hoc query.

4.14. Organization?

[kkw-from a reporting standpoint there needs to be some concept like organization or system. without this, there is no way to produce result reports that can be acted upon to provide the insight or accountability that almost all continuous monitoring instances are trying to achieve. from a scoring or grading standpoint, an endpoint needs to be associated with exactly one organization or system. it can have a many to many relationship with other types of results reporting "bins". is this important to include here? we had organization as a core asset type for this reason, so i think it is a key information element. but i also know that i do not want to define all the different reporting types, so i am unsure.]

[cek-I had not thought of this at all. Would it make sense to treat the organization and the bins as part of the guidance for creating reports? Maybe not. We should discuss.]

4.15. Guidance

[jmf- the guidance sections need more detail. . .]

[cek - What is missing? We would welcome a critique or text.]
Guidance is generally configurable by human administrators.

4.15.1. Internal Collection Guidance

An internal collector may need guidance to govern what it collects and when.

4.15.2. External Collection Guidance

An external collector may need guidance to govern what it collects and when.

4.15.3. Evaluation Guidance

An evaluator typically needs Evaluation Guidance to govern what it considers to be a good or bad security posture.

4.15.4. Retention Guidance

A SACM deployment may retain posture attributes, events, or evaluation results for some time. Retention supports ad hoc reporting and other use cases.

If information is retained, retention guidance controls what is retained and for how long.

If two or more pieces of retention guidance apply to a piece of information, the guidance calling for the longest retention should take precedence.

4.15.5. Reporting Guidance

A Report Generator typically needs Reporting Guidance to govern the reports it generates.

4.16. Provenance of Information

Each Endpoint Attribute Assertion and Report needs to be labeled with its provenance.

4.17. Endpoint

See the definition in the SACM Terminology for Security Assessment [I-D.ietf-sacm-terminology].

In the model, an endpoint can be part of another endpoint. This covers cases where multiple physical endpoints act as one endpoint.
The constituent endpoints may not be distinguishable by external observation of network behavior.

For example, a hosting center may maintain a redundant set (redundancy group) of multi-chassis setups to provide active redundancy and load distribution on network paths to WAN gateways. Multi-chassis link aggregation groups make the chassis appear as one endpoint. Traditional security controls must be applied either to physical endpoints or the redundancy groups they compose (and occasionally both). Loss of redundancy is difficult to detect or mitigate without specific posture information about the current state of redundancy groups. Even if a physical endpoint (e.g. router) that is part of a redundancy group is replaced, the redundancy group can remain the same.

4.17.1. Endpoint Identity

An endpoint identity provides both identification and authentication of the endpoint. For example, an identity may be an X.509 certificate [RFC5280] and corresponding private key. [jmf- this example should be formatted like the other examples in this section]

Not all kinds of identities are guaranteed to be unique.

4.17.2. Software Component

An endpoint contains and runs software components.

Some of the software components are assets. "Asset" is defined in RFC4949 [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."

An examination of software needs to consider both (a) software assets and (b) software that may do harm. A posture attribute collector may not know (a) from (b). It is useful to define Software Component as the union of (a) and (b).

Examples of Software Assets:

- An application
- A patch
- The operating system kernel
- A boot loader
Examples of harmful software components:

- Firmware that controls a disk drive
- A piece of JavaScript found in a web page the user visits
- A malicious entertainment app
- A malicious executable
- A web page that contains malicious JavaScript
- A business application that shipped with a virus

4.17.2.1. Unique Software Identifier

Organizations need to be able to uniquely identify and label software installed or run 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.

A piece of software may have a unique identifier, such as a SWID tag (ISO/IEC 19770).

4.18. User

4.18.1. User Identity

An endpoint is often – but not always – associated with one or more users.

A user’s identity provides both identification and authentication of the user. Eh?

5. SACM Usage Scenario Example

TODO: this section needs to refer out to wherever the operations / generalized workflow content ends up

TODO: revise to eliminate graph references

This section illustrates the proposed SACM Information Model as applied to SACM Usage Scenario 2.2.3, Detection of Posture Deviations
The following subsections describe the elements (components and elements), graph model, and operations (sample workflow) required to support the Detection of Posture Deviations scenario.

The Detection of Posture Deviations scenario involves multiple elements interacting to accomplish the goals of the scenario. Figure 1 illustrates those elements along with their major communication paths.

5.1. Graph Model for Detection of Posture Deviation

The following subsections contain examples of identifiers and metadata which would enable detection of posture deviation. These lists are by no means exhaustive - many other types of metadata would be enumerated in a data model that fully addressed this usage scenario.

5.1.1. Components

The proposed SACM Information Model contains three components, as defined in the SACM Architecture [I-D.ietf-sacm-architecture]: Posture Attribute Information Provider, Posture Attribute Information Consumer, and Control Plane.

In this example, the components are instantiated as follows:

- The Posture Attribute Information Provider is an endpoint security service which monitors the compliance state of the endpoint and reports any deviations from the expected posture.
- The Posture Attribute Information Consumer is an analytics engine which absorbs information from around the network and generates a "heat map" of which areas in the network are seeing unusually high rates of posture deviations.
- The Control Plane is a security automation broker which receives subscription requests from the analytics engine and authorizes access to appropriate information from the endpoint security service.

5.1.2. Identifiers

To represent the elements listed above, the set of identifiers might include (but is not limited to):

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

To characterize the elements listed above, the set of metadata types might include (but is not limited to):

- Authorization metadata attached to an identity identifier, or to a link between a network session identifier and an identity identifier, or to a link between a network session identifier and an address identifier.

- Location metadata attached to a link between a network session identifier and an address identifier.

- Event metadata attached to an address identifier or an identity identifier of an endpoint, which would be made available to interested parties at the time of publication, but not stored long-term. For example, when a user disables required security software, an internal collector associated with an endpoint security service might publish guidance violation event metadata attached to the identity identifier of the endpoint, to notify consumers of the change in endpoint state.

- Posture attribute metadata attached to an identity identifier of an endpoint. For example, when required security software is not running, an internal collector associated with an endpoint security service might publish posture attribute metadata attached to the identity identifier of the endpoint, to notify consumers of the current state of the endpoint.
5.1.4. Relationships between Identifiers and Metadata

Interaction between multiple sets of identifiers and metadata lead to some fairly common patterns, or "constellations", of metadata. For example, an authenticated-session metadata constellation might include a central network session with authorizations and location attached, and links to a user identity, an endpoint identity, a MAC address, an IP address, and the identity of the policy server that authorized the session, for the duration of the network session.

These constellations may be independent of each other, or one constellation may be connected to another. For example, an authenticated-session metadata constellation may be created when a user connects an endpoint to the network; separately, an endpoint-posture metadata constellation may be created when an endpoint security system and other collectors gather and publish posture information related to an endpoint. These two constellations are not necessarily connected to each other, but may be joined if the component publishing the authenticated-session metadata constellation is able to link the network session identifier to the identity identifier of the endpoint.

5.2. Workflow

The workflow for exchange of information supporting detection of posture deviation, using a standard publish/subscribe/query transport model such as available with IF-MAP [TNC-IF-MAP-SOAP-Binding] or XMPP-Grid [I-D.salowey-sacm-xmpp-grid], is as follows:

1. The analytics engine (Posture Assessment Information Consumer) establishes connectivity and authorization with the transport fabric, and subscribes to updates on posture deviations.

2. The endpoint security service (Posture Assessment Information Provider) requests connection to the transport fabric.

3. Transport fabric authenticates and establishes authorized privileges (e.g. privilege to publish and/or subscribe to security data) for the requesting components.

4. The endpoint security service evaluates the endpoint, detects posture deviation, and publishes information on the posture deviation.

5. The transport fabric notifies the analytics engine, based on its subscription of the new posture deviation information.
Other components, such as access control policy servers or remediation systems, may also consume the posture deviation information provided by the endpoint security service.

6. 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, Scott Pope, Charles Schmidt, and Steve Venema have contributed text to many sections of this document.

6.1. Contributors

The RFC guidelines no longer allow RFCs to be published with a large number of authors. Some additional authors contributed to specific sections of this document; their names are listed in the individual section headings as well as alphabetically listed with their affiliations below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Henk Birkholz</td>
<td>Fraunhofer SIT</td>
<td><a href="mailto:henk.birkholz@sit.fraunhofer.de">henk.birkholz@sit.fraunhofer.de</a></td>
</tr>
</tbody>
</table>

7. IANA Considerations

This memo includes no request to IANA.

8. 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.

The TNC IF-MAP Binding for SOAP [TNC-IF-MAP-SOAP-Binding] and TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA] document security considerations for sharing information via security automation. Most, and possibly all, of these considerations also apply to information shared via this proposed information model.

9. References

9.1. Normative References


9.2. Informative References


[I-D.ietf-sacm-architecture]
Cam-Winget, N., Ford, B., Lorenzin, L., McDonald, I., and l.loxxy@cisco.com, "Secure Automation and Continuous Monitoring (SACM) Architecture", draft-ietf-sacm-architecture-00 (work in progress), October 2014.

[I-D.ietf-sacm-requirements]

[I-D.ietf-sacm-terminology]

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

[I-D.salowey-sacm-xmpp-grid]

[IM-LIAISON-STATEMENT-NIST]

[ISO.18180]

[ISO.19770-2]
[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,

[NISTIR-7693]
Wunder, J., Halbardier, A., and D. Waltermire,
"Specification for Asset Identification 1.1", NISTIR 7693,
NISTIR-7693.pdf>.

[NISTIR-7694]
Halbardier, A., Waltermire, D., and M. Johnson,
"Specification for the Asset Reporting Format 1.1", NISTIR
NISTIR-7694.pdf>.

[NISTIR-7695]
Cheikes, B., Waltermire, D., and K. Scarfone, "Common
Platform Enumeration: Naming Specification Version 2.3",
NISTIR-7695-CPE-Naming.pdf>.

[NISTIR-7696]
Parmelee, M., Booth, H., Waltermire, D., and K. Scarfone,
"Common Platform Enumeration: Name Matching Specification
NISTIR-7696-CPE-Matching.pdf>.

[NISTIR-7697]
Cichonski, P., Waltermire, D., and K. Scarfone, "Common
Platform Enumeration: Dictionary Specification Version

[NISTIR-7698]
Waltermire, D., Cichonski, P., and K. Scarfone, "Common
Platform Enumeration: Applicability Language Specification
NISTIR-7698-CPE-Language.pdf>.


[TNC-Architecture]

[TNC-IF-M-TLV-Binding]

[TNC-IF-MAP-ICS-METADATA]

[TNC-IF-MAP-NETSEC-METADATA]

[TNC-IF-MAP-SOAP-Binding]

[TNC-IF-T-TLS]

[TNC-IF-T-Tunneled-EAP]

[TNC-IF-TNCCS-TLV-Binding]

[UML]

[W3C.REC-rdf11-concepts-20140225]
Appendix A. Security Automation with TNC IF-MAP

A.1. What is Trusted Network Connect?

Trusted Network Connect (TNC) is a vendor-neutral open architecture [TNC-Architecture] and a set of open standards for network security developed by the Trusted Computing Group (TCG). TNC standards integrate security components across end user systems, servers, and network infrastructure devices into an intelligent, responsive, coordinated defense. TNC standards have been widely adopted by vendors and customers; the TNC endpoint assessment protocols [TNC-IF-M-TLV-Binding] [TNC-IF-TNCCS-TLV-Binding] [TNC-IF-T-Tunneled-EAP] [TNC-IF-T-TLS] were used as the base for the IETF NEA RFCs [RFC5792] [RFC5793] [RFC7171] [RFC6876].

Traditional information security architectures have separate silos for endpoint security, network security, server security, physical security, etc. The TNC architecture enables the integration and categorization of security telemetry sources via the information model contained in its Interface for Metadata Access Points (IF-MAP) [TNC-IF-MAP-SOAP-Binding]. IF-MAP provides a query-able repository of security telemetry that may be used for storage or retrieval of such data by multiple types of security systems and endpoints on a vendor-neutral basis. The information model underlying the IF-MAP repository covers, directly or indirectly, all of the security information types required to serve SACM use-cases.

A.2. What is TNC IF-MAP?

IF-MAP provides a standard client-server protocol for MAP clients to exchange security-relevant information via database server known as the Metadata Access Point or MAP. The data (known as "metadata") stored in the MAP is XML data. Each piece of metadata is tagged with a metadata type that indicates the meaning of the metadata and
identifies an XML schema for it. Due to the XML language, the set of metadata types is easily extensible.

The MAP is a graph database, not a relational database. Metadata can be associated with an identifier (e.g. the email address "user@example.com") or with a link between two identifiers (e.g. the link between MAC address 00:11:22:33:44:55 and IPv4 address 192.0.2.1) where the link defines an association (for example: a relation or state) between the identifiers. These links between pairs of identifiers create an ad hoc graph of relationships between identifiers. The emergent structure of this graph reflects a continuously evolving knowledge base of security-related metadata that is shared between various providers and consumers.

A.3. What is the TNC Information Model?

The TNC Information Model underlying IF-MAP relies on the graph database architecture to enable a (potentially distributed) MAP service to act as a shared clearinghouse for information that infrastructure devices can act upon. The IF-MAP operations and metadata schema specifications (TNC IF-MAP Binding for SOAP [TNC-IF-MAP-SOAP-Binding], TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA], and TNC IF-MAP Metadata for ICS Security [TNC-IF-MAP-ICS-METADATA]) define an extensible set of identifiers and data types.

Each IF-MAP client may interact with the IF-MAP graph data store through three fundamental types of operation requests:

- Publish, which may create, modify, or delete metadata associated with one or more identifiers and/or links in the graph
- Search, which retrieves a selected sub-graph according to a set of search criteria
- Subscribe, which allows a client to manage a set of search commands which asynchronously return selected sub-graphs when changes to that sub-graph are made by other IF-MAP clients

The reader is invited to review the existing IF-MAP specification [TNC-IF-MAP-SOAP-Binding] for more details on the above graph data store operation requests and their associated arguments.

The current IF-MAP specification provides a SOAP [W3C.REC-soap12-part1-20070427] binding for the above operations, as well as associated SOAP operations for managing sessions, error handling, etc.
Appendix B. Text for Possible Inclusion in the Terminology Draft

B.1. 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.

B.1.1. 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.

B.1.2. 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.

Appendix C. Text for Possible Inclusion in the Architecture or Use Cases

C.1. Introduction

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

A system administrator needs to be able to determine which elements of 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.
C.2. Core Principles

This information model is built on the following core principles:

- Collection and Evaluation are separate tasks.
- 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.
- Every entity (human or machine) that notifies, queries, or responds to any guidance, collection, or evaluator 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.
- Policies are reflected in guidance for collection, evaluation, and reporting.
- 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.
- 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.
- 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.

C.3. 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
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:

- **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.

- **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.

- **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.

- **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 2, Consumers either request information from, or are notified by, Producers.

```
+----------+     Request     +----------+
|          <-----------------+          |
| Producer |                 | Consumer |
|          +----------------->          |
+----------+    Response     +----------+
+----------+                 +----------+
|          |     Notify      |          |
| Producer +-----------------> Consumer |
|          |                 |          |
+----------+                 +----------+
```

*Figure 2: Example Producer/Consumer Interactions*
As illustrated in Figure 3, writing and querying from data repositories are a way in which this interaction can occur in an asynchronous fashion.

```
+----------+                +----------+
|          |                |          |
| Producer |                | Consumer |
+----------+                +----------+
        |                          |
Write   |     +------------+       | Query
        |            |       |
+-----> Repository +-------+
        |            |
```

Figure 3: 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:

- the collection capability can reside on the endpoint or not.
- the collection producer can be part of the collection capability or not.
- a repository can be directly associated with a producer and/or an evaluator or stand on its own.
- there can be multiple "levels" of producers and consumers.
This illustrative example in Figure 4 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.

Appendix D. Text for Possible Inclusion in the Requirements Draft

D.1. 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.

D.2. Problem Scope

The goal of this iteration of the information model is to define the information needs for an organization to effectively monitor 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 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 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.

Appendix E. Text With No Clear Home Yet

E.1. Operations

Operations that may be carried out the proposed SACM Information Model are:

- Publish data: Security information is made available in the information model when a component publishes data to it.

- Subscribe to data: A component seeking to consume an on-going stream of security information "subscribes" to such data from the information model.

- Query: This operation enables a component to request a specific set of security data regarding a specific asset (such as a specific user endpoint).

The subscribe capability will allow SACM components to monitor for selected security-related changes in the graph data store without incurring the performance penalties associated with polling for such changes.

E.1.1. Generalized Workflow

The proposed SACM Information Model would be most commonly used with a suitable transport protocol for collecting and distributing security data across appropriate network platforms and endpoints. The information model is transport agnostic and can be used with its native transport provided by IF-MAP or by other data transport protocols such as the recently proposed XMPP-Grid.

1. A Posture Assessment Information Consumer (Consumer) establishes connectivity and authorization with the transport fabric.

2. A Posture Assessment Information Provider (Provider) with a source of security data requests connection to the transport fabric.
3. Transport fabric authenticates and establishes authorized privileges (e.g. privilege to publish and/or subscribe to security data) for the requesting components.

4. Components may either publish security data, subscribe to security data, query for security data, or any combination of these operations.

Any component sharing information - either as Provider or Consumer - may do so on a one-to-one, one-to-many and/or many-to-many meshed basis.

E.2. 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.

E.3. Information Model Elements

TODO: Kim to pull up relevant content into section 4 / 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. Appendix E.3.1 Asset Identifiers: Describes identification of many different asset types including endpoints.

2. Endpoint Characterization
   A. Appendix E.3.3 Endpoint characterization: This directly maps to the requested area.

3. Endpoint Attribute Expression/Representation
   A. Appendix E.3.4 Posture Attribute Expression: This corresponds to the first part of "Endpoint Attribute Expression/Representation."
   B. Appendix E.3.5 Actual Value Representation: This corresponds to the second part of "Endpoint Attribute Expression/Representation."

4. Policy evaluation expression and results reporting
   A. Appendix E.3.6 Evaluation Guidance: This corresponds to the first part of "Policy evaluation expression and results reporting."
   B. Appendix E.3.7 Evaluation Result Reporting: corresponds to the second part of "Policy evaluation expression and results reporting."

Additionally, Appendix E.3.2 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.

E.3.1. 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.

E.3.1.1. Related Work
E.3.1.1.1. 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 5), 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 5: Asset Identification Class Hierarchy
This table presents a mapping of notional SACM asset types to those asset types provided by the Asset Identification specification.

<table>
<thead>
<tr>
<th>SACM Asset Type</th>
<th>Asset Identification Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>computing-device</td>
<td>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.</td>
</tr>
<tr>
<td>Hardware</td>
<td>Not Applicable</td>
<td>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.</td>
</tr>
<tr>
<td>Software</td>
<td>software</td>
<td>Direct mapping.</td>
</tr>
<tr>
<td>Network</td>
<td>network</td>
<td>Direct mapping.</td>
</tr>
<tr>
<td>Organization</td>
<td>organization</td>
<td>Direct mapping.</td>
</tr>
<tr>
<td>Person</td>
<td>person</td>
<td>Direct mapping.</td>
</tr>
</tbody>
</table>

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.

E.3.1.2. 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.
E.3.1.2.1. Related Work

The previously introduced asset identification specification (see Appendix E.3.1.1.1 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.

E.3.1.3. 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:

- 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.
- Indicate the presence of the software unit on a given endpoint.
- To determine what endpoints are the targets for an assessment based on what software is installed on that endpoint.
- 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.

E.3.1.3.1. 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.
The Common Platform Enumeration (CPE) [CPE-WEBSITE] is composed of a family of four specifications 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:

- 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.
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.

E.3.1.3.1.1.2. 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 Appendix E.3.1.1.)

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.

E.3.1.3.1.2. 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:

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

E.3.1.4. 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.

E.3.2. 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).

E.3.2.1. 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).

E.3.2.1.1. Related Work

E.3.2.1.1.1. 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.
E.3.2.1.1.2. Open Vulnerability and Assessment Language

E.3.2.1.1.2.1. 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.

![OVAL Data Model Diagram](image)

Note: The direction of the arrows indicate a model dependency

**Figure 6: The OVAL Data Model**

The OVAL data model [OVAL-LANGUAGE], visualized in Figure 6, is composed of a number of different components. The components are:

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

- **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.

- Variables: Constructs that allow for the parameterization of the elements used in the Definitions component based on externally provided values.

- 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 elements 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.

- 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.

- 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:

- 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.

- 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.

- 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.

- 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.

E.3.2.1.1.2.2. 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.

E.3.2.2. 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 Appendix E.3.2.1) 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.

E.3.2.2.1. Related Work

E.3.2.2.1.1. 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.

E.3.2.2.1.2. 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:

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

E.3.2.3. 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.

E.3.2.3.1. Related Work

E.3.2.3.1.1. 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.

E.3.3. 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:

- A manual input process and entered into records associated with the endpoint, or
- 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.

E.3.3.1. Related Work

E.3.3.1.1. Extensible Configuration Checklist Description Format

E.3.3.1.1.1. 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.

E.3.3.1.1.2. 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.

E.3.3.1.2. Asset Reporting Format

E.3.3.1.2.1. 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:

- **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.

- **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.

- **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.

- **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.

E.3.3.1.2.2. 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 Appendix E.3.3.1.2.3) 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 Appendix E.3.1.1.1) 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.

E.3.3.1.2.3. 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 Appendix E.3.3.1.2.3, the Asset Identification specification is included within the Asset Reporting Framework (ARF) and therefore can be used in concert with that specification as well.

E.3.3.1.3. 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.

E.3.4. 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.

E.3.4.1. 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.

E.3.4.2. 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.

E.3.4.2.1. Related Work

E.3.4.2.1.1. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix E.3.2.1.1.2.1. 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.

E.3.4.2.1.2. 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 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.

E.3.4.2.1.3. 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.

E.3.5. 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.

E.3.5.1. 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.

E.3.5.1.1. Related Work

E.3.5.1.1.1. 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.

E.3.5.1.1.2. Software Identification Tags

SWID tag were previously introduced in Appendix E.3.1.3.1.2. 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.

E.3.5.2. 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.

E.3.5.2.1. Related Work

E.3.5.2.1.1. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix E.3.2.1.1.2.1. 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.

E.3.5.2.1.2. NETCONF-Based Collection

Introduced earlier in Appendix E.3.4.2.1.2, 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
Internet-Draft           SACM Information Model            February 2015

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 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.

E.3.5.2.1.3.  SNMP-Based Collection

The SNMP, previously introduced in Appendix E.3.4.2.1.3, 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 useful to consider this
protocol as a standardized mechanism for collecting posture attribute
data from endpoints in an enterprise.

E.3.6.  Evaluation Guidance

E.3.6.1.  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:

- configuration item identifiers,
- platform configuration identifiers, and
- collected Platform Configuration Posture Attributes.
E.3.6.1.1. Related Work

E.3.6.1.1.1. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix E.3.2.1.1.2.1. 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, which 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.

E.3.6.1.1.2. XCCDF Rule

A general description of XCCDF was provided in Appendix E.3.3.1.1.1. 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 Appendix E.3.3.1.3), 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.

E.3.7. Evaluation Result Reporting

E.3.7.1. 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:

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

E.3.7.1.1. Related Work

E.3.7.1.1.1. XCCDF TestResults

A general description of the eXtensible Configuration Checklist Description Format (XCCDF) was provided in section Appendix E.3.3.1.1.1. 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:

- The identity of the assessed endpoint. See Appendix E.3.3.1.1.2 for more about XCCDF structures used for endpoint identification.
- Any tailoring of the checklist that might have been employed. See Appendix E.3.6.1.1.2 for more on how XCCDF supports tailoring.
- The individual results of the assessment of each enabled XCCDF Rule in the checklist. See Appendix E.3.6.1.1.2 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.

E.3.7.1.1.2. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix E.3.2.1.1.2.1. 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.
**E.3.7.1.1.3. Asset Summary Reporting**

A general overview of ASR was provided previously in Appendix E.3.5.1.1.1. 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.

**E.3.7.1.1.4. ARF**

A general overview of ARF was provided previously in Appendix E.3.3.1.2.1. 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.

**E.3.7.2. 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.

**Appendix F. Graph Model**

TODO: Write text on how the information model above can be realized in this kind of graph model.

The graph model describes how security information is structured, related, and accessed. Control of operations to supply and/or access the data is architecturally distinct from the structuring of the data in the information model. Authorization may be applied by the Control Plane (as defined in the SACM Architecture [I-D.ietf-sacm-architecture]) to requests for information from a consumer or requests for publication from a provider, and may also be applied by a provider to a direct request from a consumer.
This architecture addresses information structure independently of
the access/transport of that information. This separation enables
scalability, customizability, and extensibility. Access to provide
or consume information is particularly suited to publish/subscribe/
query data transport and data access control models.

This graph model is a framework that:

- Facilitates the definition of extensible data types that support
  SACM’s use cases
- Provides a structure for the defined data types to be exchanged
  via a variety of data transport models
- Describes components used in information exchange, and the objects
  exchanged
- Captures and organizes evolving information and information
  relationships for multiple data publishers
- Provides access to the published information via publish, query,
  and subscribe operations
- Leverages the knowledge and experience gained from incorporating
  TNC IF-MAP into many disparate products that have to integrate and
  share an information model in a scalable, extensible manner

F.1. Background: Graph Models

Knowledge is often represented with graph-based formalisms. A common
formalism defines a graph as follows:

- A set of *vertices*
- A set of *edges*, each connecting two vertices (technically, an
edge is an ordered pair of vertices)
- A set of zero or more *properties* attached to each vertices and
  edges. Each property consists of a type and a optionally a value.
The type and the value are typically just strings
A pair of vertices connected by an edge is commonly referred to as a triple that comprises subject, predicate and object. For example, subject = Sue Wong, predicate = is-parent-of, object = Ann Wong. A common language that uses this representation is the Resource Description Framework (RDF) [W3C.REC-rdf11-concepts-20140225].

F.2. Graph Model Overview

The proposed model, influenced by IF-MAP, is a labeled graph: each vertex has a label.

A table of synonyms follows.

<table>
<thead>
<tr>
<th>Graph Theory</th>
<th>Graph Databases</th>
<th>IF-MAP and This Internet Draft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex or Node</td>
<td>Node</td>
<td>-</td>
</tr>
<tr>
<td>Label</td>
<td>-</td>
<td>Identifier</td>
</tr>
<tr>
<td>Edge</td>
<td>Edge</td>
<td>Link</td>
</tr>
<tr>
<td>-</td>
<td>Property</td>
<td>Metadata Item</td>
</tr>
</tbody>
</table>

In this mode, identifiers and metadata have hierarchical structure.

The graphical aspect makes this model well suited to non-hierarchical relationships, such as connectivity in a computer network.

Hierarchical properties allow this model to accommodate structures such as YANG [RFC6020] data models.

F.3. Identifiers

Each identifier is an XML element. For extensibility, schemas use xsd:anyAttribute and such.

Alternately, this model could be changed to use another hierarchical notation, such as JSON.
Identifiers are unique: two different vertices cannot have equivalent identifiers.

An identifier has a type. There is a finite, but extensible, set of identifier types. If the identifier is XML, the type is based on the XML schema.

In IF-MAP, standard identifier types include IP address, MAC address, identity, and overlay network. Additional identifier types will need to be standardized for SACM use cases.

Any number of metadata items can be attached to an identifier.

Some identifiers, especially those relating to identity, address, and location, require the ability to specify an administrative domain (such as AD domain, L2 broadcast domain / L3 routing domain, or geographic domain) in order to differentiate between instances with the same name occurring in different realms.

F.4. Links

A link can be thought of as an ordered pair of identifiers.

Any number of metadata items can be attached to a link.

F.5. Metadata

A metadata item is the basic unit of information, and is attached to an identifier or to a link.

A given metadata item is an XML document. In IF-MAP metadata items are generally small. However, larger ones, such as YANG data models, can also fit. For extensibility, the XML schemas of metadata items use xsd:anyAttribute and such.

Alternately, this model could be changed to use another hierarchical notation, such as JSON.

A metadata item has a type. There is a finite, but extensible, set of metadata types. If the metadata item is XML, the type is based on the XML schema. An example metadata type is http://www.trustedcomputinggroup.org/2010/IFMAP-METADATA/2#device-characteristic.

F.6. Use for SACM

Many of the information elements can be represented as vertices, and many of the relationships can be represented as edges.

Identifiers are like database keys. For example, there would be identifiers for addresses, identities, unique endpoint identifiers, software component identifiers, and hardware component identifiers. The inventory of software instances and hardware instances within an endpoint might be expressed using a single YANG description, as a single metadata item in the graph. Where to put Endpoint Attribute Assertions, Evaluation Results, and the like is an open question.

F.7. Provenance

Provenance helps to protect the SACM ecosystem against a misled or malicious provider.

The provenance of a metadata item includes:

- The time when the item was produced
- The component that produced the item, including its software and version
- The policies that governed the producing component, with versions
- The method used to produce the information (e.g., vulnerability scan)

How provenance should be expressed is an open question. For reference, in IF-MAP provenance of a metadata item is expressed within the metadata item [TNC-IF-MAP-NETSEC-METADATA]. For example, there is a top-level XML attribute called "timestamp".

It is critical that provenance be secure from tampering. How to achieve that security is out of scope of this document.

F.8. Extensibility

Anyone can define an identifier type or a metadata type, by creating an XML schema (or other specification). There is no need for a central authority. Some deployments may exercise administrative control over the permitted identifier types and metadata types; others may leave components free rein.

A community of components can agree on and use new identifier and metadata types, if the administrators allow it. This allows rapid
innovation. Intermediate software that conveys graph changes from one component to another does not need changes. Components that do not understand the new types do not need changes. Accordingly, a consumer normally ignores metadata types and identifier types it does not understand.

As a proof point for this agility, the original use cases for TNC IF-MAP Binding for SOAP [TNC-IF-MAP-SOAP-Binding] were addressed in TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA]. Some years later an additional, major set of use cases, TNC IF-MAP Metadata for ICS [TNC-IF-MAP-ICS-METADATA], were specified and implemented.

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

Clifford Kahn
Pulse Secure, LLC
2700 Zanker Road, Suite 200
San Jose, CA 95134
USA

Email: cliffordk@pulsesecure.net
Abstract

This document defines the scope and set of requirements for the Secure Automation and Continuous Monitoring (SACM) architecture, data model and protocols. The requirements and scope are based on the agreed upon use cases.

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 September 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Today's environment of rapidly-evolving security threats highlights the need to automate the sharing of such information while protecting user information as well as the systems that store, process, and transmit this information. Security threats can be detected in a number of ways. SACM's charter focuses on how to collect and share this information based on use cases that involve posture assessment of endpoints.

Scalable and sustainable collection, expression, and evaluation of endpoint information is foundational to SACM's objectives. To secure and defend a 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 document focuses on describing the requirements for facilitating the exchange of posture assessment information, in particular, for the use cases as exemplified in [I-D.ietf-sacm-use-cases]. Also, this document uses terminology defined in [I-D.ietf-sacm-terminology].
2. Requirements

This document defines requirements based on the SACM use cases defined in [I-D.ietf-sacm-use-cases]. This section describes the requirements used by SACM to assess and compare candidate data models, interfaces, and protocols, to suit the SACM architecture. These requirements express characteristics or features that a candidate protocol or data model must be capable of offering to ensure security and interoperability.

In order to address the needs for determining, sharing, and using posture information, the following tasks should be considered:

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. Determine the expected values for the data elements and attributes that 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.

2.1. Requirements for SACM

Many deployment scenarios can be instantiated to address the above tasks and use cases defined in [I-D.ietf-sacm-use-cases]. To ensure interoperability, scalability, and flexibility in any of these deployments, the following requirements are defined for proposed SACM standards:

G-001 Solution Extensibility: The data models, protocols, and transports defined by SACM MUST be extensible to allow support for non-standard and future extensions.
1. The transport protocol MUST support the ability to add new operations while maintaining backwards compatibility.

2. The query language MUST allow for general inquiries, as well as expression of specific paths to follow; the retrieval of specific information based on an event, as well as on a continuous basis and the ability to retrieve specific pieces of information, specific types or classes of information, and/or the entirety of available information.

3. The information model MUST accommodate the addition of new data types and/or schemas in a backwards compatible fashion.

G-002 Interoperability: The data models, protocols, and transports must be specified with enough details to ensure interoperability. [EDITOR Note: Can we remove this requirement?]

G-003 Scalability: The data models, protocols, and transports MUST be scalable. SACM must support a broad set of deployment scenarios. Scalability must be addressed to support:

* Large datagrams: It is possible that the size of posture assessment information can vary from a single assessment that is small in (record or datagram) size to a very large datagram or a very large set of assessments.

* Large number of providers and consumers: A deployment may consist of a very large number of endpoints requiring or producing (or both) posture assessment information.

* Large number of target endpoints: A deployment may be managing information of a very large number of target endpoints.

G-004 Agility: The data model, protocols, and transports MUST be suitably specified to enable implementations to fit into the different deployment models and scenarios, including considerations for lightweight implementations of data models and transports.

G-005 Information Extensibility: A method for expressing both standard and non-standard (implementation-specific) data attributes while avoiding collisions SHOULD be defined. For interoperability and scope boundary, an explicit set of data attributes MUST be defined as mandatory to implement.

G-006 Data Integrity: A method for ensuring data integrity MUST be provided. This method is required to be available (i.e. all data-
handling components must support it), but is not required to be used in all cases.

G-007 Data Isolation: A method for partitioning data MUST be supported to accommodate considerations such as geographic, regulatory, overlay boundaries, and federation, where an organization may want to differentiate between information that can be shared outside its own domain and information that cannot. As with the requirement for data integrity, this method is required to be available (i.e. all data-handling components must support it), but is not required to be used in all cases.

G-008 Versioning and Backward Compatibility: Announcement and negotiation of versions, inclusive of existing capabilities (such as transport protocols, data models, specific attributes within data models, standard attribute expression sets, etc.) MUST be supported. Negotiation for both versioning and capability is needed to accommodate future growth and ecosystems with mixed capabilities.

G-009 Discovery: There MUST be a mechanism for components to discover what information is available across the ecosystem (i.e. a method for cataloging data available in the ecosystem and advertising it to consumers), and where to go to get a specific piece of that information. For example, providing a method by which a node can locate the advertised information so that consumers are not required to have a priori knowledge to find available information.

G-010 Endpoint Discovery: SACM MUST define the means by which endpoints may be discovered. Use Case 2.1.2 describes the need to discover endpoints and their composition.

G-011 Unsolicited publication, updates, or change modifications with filtering MUST be defined: Use Cases 2.1.3, 2.1.4 and 2.1.5 describe the need for the data model to support the means for the information to be published unsolicited. Similarly, the data model must support the means for a requestor to obtain updates or change modifications without constant querying (or polling). Like the query operation, these update notifications must be able to be set up with a filter to allow for only a subset of posture assessment information to be obtained.

G-012 Device Interface: the interfaces by which endpoint communicate to share endpoint posture information MUST be well defined.

G-013 Device location and network topology: the SACM architecture and interfaces MUST allow for the endpoint (network) location and network topology to be modeled and understood. Where appropriate,
the data model and the interfaces SHOULD allow for discovery of endpoint location or network topology or both.

G-014 Target Endpoint Identity: the SACM architecture and interfaces MUST support the ability of components to provide attributes that can be used to compose an identity for a target endpoint. These identities MAY be composed of attributes from one or more SACM components.

2.2. Requirements for the Architecture

At the simplest abstraction, the SACM architecture represents the core components and interfaces needed to perform the production and consumption of posture assessment information. Requirements relating to the SACM’s architecture include:

ARCH-001 Scalability: The architectural components MUST account for a range of deployments, from very small sets of endpoints to very large deployments.

ARCH-002 Flexibility: The architectural components MUST account for different deployment scenarios where the architectural components may be implemented, deployed, or used within a single application, service, or network, or may comprise a federated system.

ARCH-003 Separation of Data and Management functions: SACM MUST define both the configuration and management of the SACM data models and protocols used to transport and share posture assessment information.

ARCH-004 Topology Flexibility: Both centralized and decentralized (peer-to-peer) information exchange MUST be supported. Centralized data exchange enables use of a common data format to bridge together data exchange between diverse systems, and can leverage a virtual data store that centralizes and offloads all data access, storage, and maintenance to a dedicated resource. Decentralized data exchange enables simplicity of sharing data between relatively uniform systems, and between small numbers of systems, especially within a single enterprise domain.

ARCH-005 Modularity: Announcement and negotiation of functional capabilities (such as authentication protocols, authorization schemes, data models, transport protocols, etc.) must be supported, enabling a SACM component to make inquiries about the capabilities of other components in the SACM ecosystem.

ARCH-006 Role-based Authorization: The SACM architecture MUST be capable of effecting role based authorization. Distinction of
endpoints capable and authorized to provide or consume information is required to address appropriate access controls.

ARCH-007 Context-based Authorization: The SACM architecture MUST be capable of effecting context based authorization. Different policies (e.g. business, regulatory, etc.) may specify what data may be exposed or shared by particular consumers, as well as how consumers may be required to share the information. The context defines the composite set of guidelines provided by policy for affecting what posture data is allowed to be communicated from providers to consumers.

2.3. Requirements for the Information Model

The SACM information represents the abstracted representation for the Posture Assessment information to be communicated. SACM data models must adhere and comply to the SACM Information Model. The requirements for the SACM information model include:

IM-001 Extensible Attribute Dictionary: the Information Model MUST define the minimum set of attributes for communicating Posture Information. The attributes should be defined within an extensible attribute dictionary to enable data models to adhere to SACM’s required attributes as well as allow for their own extensions.

IM-002 Ephemerality: The Information Model SHOULD account for the Posture information’s ephemeral nature as the data may be provided by a requestor either solicited or unsolicited.

IM-003 Data model negotiation: SACM’s Information Model MUST allow support for different data models and data model versions. The SACM Information Model MUST include the ability to discover and negotiate the use of a particular or any data model.

2.4. Requirements for the Data Model

The SACM information model represents an abstraction for "what" information can be communicated and "how" it is to be represented and shared.

It is expected that as applications may produce posture assessment information, they may share it using a specific data model. Similarly, applications consuming or requesting posture assessment information, may require it be based on a specific data model. Thus, while there may exist different data models and schemas, they should adhere to the SACM information model and meets the requirements defined in this section.
The specific requirements for candidate data models include:

DM-001 The data model MUST define the data attributes as objects that MUST be uniquely referenced (e.g. endpoint, IP address, asset).

DM-002 The data model MAY be structured into modules and submodules to allow for data references within a module. For example, an endpoint may be defined as a module that references one or more submodules that further describe the one or more assets. Constraints and interfaces may further be defined to resolve or tolerate ambiguity in the references (e.g. same IP address used in two separate networks).

DM-003 The interfaces and actions in the data model MUST include support for rootless searches and wildcard searches.

DM-004 The search interfaces and actions MUST include the ability to start a search anywhere within a data model structure.

DM-005 The data model SHOULD include management of the data, including data lifetime management (longevity or expiration of data) and persistence considerations.

DM-006 Ephemerality: The data model SHOULD include the ability to allow providers of data to provide the data as a whole or when updates occur. The data may be provided by a requestor either solicited or unsolicited.

DM-007 The data model SHOULD allow for a loose coupling between the provider and the requestor.

DM-008 The interfaces and actions in the data model MUST include the ability to identify data from a specific provider.

DM-009 Data cardinality: The data model MUST describe their constraints (e.g. cardinality). As posture information and the tasks for collection, aggregation or evaluation, could comprise one or more attributes, interfaces and actions MUST allow and account for such cardinality as well as whether the attributes are conditional, optional, or mandatory.

DM-010 Data model negotiation: The interfaces and actions in the data model MUST include capability negotiation to enable discovery of supported and available data types and schemas.

DM-011 Provenance: The data model MUST include the ability for providers to identify the data origin and provide a method for provenance information to be captured and communicated.
DM-012  Attribute Dictionary: Use Cases in the whole of Section 2 describe the need for an attribute dictionary. With SACM’s scope focused on posture assessment, the data model attribute collection and aggregation MUST have a well-understood set of attributes inclusive of their meaning or usage intent.

DM-013  Transport Agnostic: the data model SHOULD be transport agnostic, to allow for the data operations to leverage the most appropriate transport Internet layer (e.g. Link Layer, TCP, UDP, etc.). [Editor’s Note: Per Dan Romanascu’s comment, should this requirement be removed? Or perhaps moved to the Data Model Operations, whereby the operations should be defined in a transport agnostic encapsulation?]

DM-014  The data model SHOULD allow the provider to include the information’s origination time.

DM-015  The data model SHOULD allow the provider to include attributes defining how the data was generated (e.g. self-reported, reported by aggregator, scan result, etc.).

DM-016  The data model SHOULD allow the provider to include attributes defining the location of the data source.

DM-017  The data model SHOULD allow the provider to include attributes defining whether the information provided is a delta, partial, or full set of information.

DM-018  The data model MUST support the collection of attributes by a variety of collectors, including internal collectors, external collectors with an authenticated relationship with the endpoint, and external collectors based on network and other observations.

2.5. Requirements for Data Model Operations

Posture information data adhering to a Data Model must also provide interfaces that include operations for access and production of the data. The specific requirements for such operations include:

OP-001  Synchronization: Request and response operations SHOULD be timestamped, and published information SHOULD capture time of publication. Actions or decisions based on time-sensitive data (such as user logon/logoff, endpoint connection/disconnection, endpoint behavior events, etc.) are all predicated on a synchronized understanding of time. A method for detecting and reporting time discrepancies SHOULD be provided.
OP-002  Collection separation: The request for a data item MUST include enough information to properly identify the item to collect, but the request shall not be a command to directly execute nor directly be applied as arguments to a command. The purpose of this requirement is primarily to reduce the potential attack vectors, but has the additional benefit of abstracting the request for collection from the collection method, thereby allowing more flexibility in how collection is implemented.

OP-003  Collection composition: A collection request MAY be composed of multiple collection requests (which yield collected values). The desire for multiple values MUST be expressed as part of the collection request, so that the aggregation can be resolved at the point of collection without having to interact with the requester.

OP-004  Attribute-based query: A query operation SHOULD be based on a set of attributes. Use Case 2.1.2 describes the need for the data model to support a query operation based on a set of attributes to facilitate collection of information such as posture assessment, inventory (of endpoints or endpoint components), and configuration checklist.

OP-005  Information-based query with filtering: The query operation MUST support filtering. Use Case 2.1.3 describes the need for the data model to support the means for the information to be collected through a query mechanism. Furthermore, the query operation requires filtering capabilities to allow for only a subset of information to be retrieved. The query operation MAY be a synchronous request or asynchronous request.

OP-006  Data model scalability: The data model MUST be scalable. Use Cases 2.1.4 and 2.1.5 describes the need for the data model to support scalability. For example, the query operation may result in a very large set of attributes, as well as a large set of targets.

OP-007  The data model MUST allow a SACM component to communicate what data was used to construct the target endpoint’s identity, so other SACM components can determine whether they are constructing an equivalent target endpoint (and their identity) and whether they have confidence in that identity. SACM components SHOULD have interfaces defined to transmit this data directly or to refer to where the information can be retrieved.

2.6. Requirements for Transport Protocols

The requirements for transport protocols include:
T-001 Transport variability: Different transports MUST be supported to address different deployment and time constraints. Supporting transports MAY be at the data link layer, network, transport, or application layers.

T-002 Data Integrity: Transport protocols MUST be able to ensure data integrity.

T-003 Data Confidentiality: Transport protocols MUST be able to support data confidentiality. Transport protocols SHOULD ensure data protection for data in transit by encryption to provide confidentiality, integrity, and robustness against protocol-based attacks. Note that while the transport MUST be able to support data confidentiality, implementations MAY choose to make confidentiality optional. Protection for data at rest is not in scope for SACM. Data protection MAY be used for both privacy and non-privacy scenarios.

T-004 Transport protection: Transport protocols MUST be capable of supporting mutual authentication and replay protection.

T-005 Transport reliability: Transport protocols MUST provide reliable delivery of data. This includes the ability to perform fragmentation and reassembly, and to detect replays.

3. Acknowledgements

The authors would like to thank Barbara Fraser, Jim Bieda, and Adam Montville for reviewing and contributing to this draft.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This document defines the requirements for SACM. As such, it is expected that several data models, protocols and transports may be defined or reused from already existing standards. This section will highlight security considerations that may apply to SACM based on the architecture and standards applied in SACM. In particular, highlights to security considerations that may apply to the SACM reference architecture and standard data models and transports will be discussed.

To address security and privacy considerations, the data model, protocols and transport must consider authorization based on consumer function and privileges, to only allow authorized consumers and
providers to access specific information being requested or published.

To enable federation across multiple entities (such as across organizational or geographic boundaries) authorization must also extend to infrastructure elements themselves, such as central controllers / brokers / data repositories.

In addition, authorization needs to extend to specific information or resources available in the environment. In other words, authorization should be based on both subject (the information requestor) and object (the information requested). The method by which this authorization is applied is unspecified.

With SACM’s charter focus on the sharing of posture information for improving efficacy of security applications such as compliance, configuration, assurance and other threat and vulnerability reporting and remediation systems. While the goal is to facilitate the flow of information securely, it is important to note that participating endpoints may not be cooperative or trustworthy.

5.1. Trust between Provider and Requestor

The information given from the provider to a requestor may come with different levels of trustworthiness given the different potential deployment scenarios and compromise either at the provider, the requestor or devices that are involved in the transport between the provider and requestor. This section will describe the different considerations that may reduce the level of trustworthiness of the information provided.

In the information transport flow, it is possible that some of the devices may serve as proxies or brokers and as such, may be able to observe the communications flowing between an information provider and requestor. Without appropriate protections

It is common to, in general, distrust the network service provider, unless the full hop by hop communications process flow is well understood. As such, the posture information provider should protect the posture information data it provides as well as the transport it uses. Similarly, while there may be providers whose goal is to openly share its information, there may also be providers whose policy is to grant access to certain posture information based on its business or regulatory policy. In those situations, a provider may require full authentication and authorization of the requestor (or set of requestors) and share only the authorized information to the authenticated and authorized requestors.
A requestor beyond distrusting the network service provider, must also account that the information received from the provider may have been communicated through an undetermined network communications system. That is, the posture information may have traversed through many devices before reaching the requestor. As such, providing non-repudiation in SACM is out of scope. However, SACM specifications should provide the means for allowing non-repudiation possible and at minimum, provide endpoint authentication and transport integrity.

A requestor may require data freshness indications, both knowledge of data origination as well as time of publication so that it can make more informed decisions about the relevance of the data based on its currency and/or age.

It is also important to note that endpoint assessment reports, especially as they may be provided by the target endpoint may pose untrustworthy information. The considerations for this is described in Section 8 of [RFC5209].

The trustworthiness of the posture information given by the provider to one or many requestors is dependent on several considerations. Some of these include the requestor requiring:

- Full disclosure of the network topology path to the provider(s).
- Direct (peer to peer) communication with the provider.
- Authentication and authorization of the provider.
- Either or both confidentiality and integrity at the transport layer.
- Either or both confidentiality and integrity at the data layer.

6. References

6.1. Normative References

[I-D.ietf-sacm-terminology]

[I-D.ietf-sacm-use-cases]
6.2. Informative References


Authors’ Addresses

Nancy Cam-Winget
Cisco Systems
3550 Cisco Way
San Jose, CA  95134
US

Email: ncamwing@cisco.com

Lisa Lorenzin
Pulse Secure
2700 Zanker Rd., Suite 200
San Jose, CA  95134
US

Email: llorenzin@pulsesecure.net
Abstract

This memo documents terminology used in the documents produced by SACM (Security Automation and Continuous Monitoring).

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 August 15, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Our goal with this document is to improve our agreement on the terminology used in documents produced by the IETF Working Group for Security Automation and Continuous Monitoring. Agreeing on terminology should help reach consensus on which problems we’re trying to solve, and propose solutions and decide which ones to use.

2. Terms and Definitions

This section describes terms that have been defined by other RFC’s and defines new ones. The predefined terms will reference the RFC and where appropriate will be annotated with the specific context by which the term is used in SACM.

Assessment

Defined in [RFC5209] as "the process of collecting posture for a set of capabilities on the endpoint (e.g., host-based firewall) such that the appropriate validators may evaluate the posture against compliance policy."
Within this document the use of the term is expanded to support other uses of collected posture (e.g. reporting, network enforcement, vulnerability detection, license management). The phrase "set of capabilities on the endpoint" includes: hardware and software installed on the endpoint."

Asset

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.

Asset characterization

Asset characterization is the process of defining attributes that describe properties of an identified asset.

Asset Management

The process by which assets are provisioned, updated, maintained and deprecated.

Asset Targeting

Asset targeting is the use of asset identification and categorization information to drive human-directed, automated decision making for data collection and analysis in support of endpoint posture assessment.

Attribute

Defined in [RFC5209] as "data element including any requisite meta-data describing an observed, expected, or the operational status of an endpoint feature (e.g., anti-virus software is currently in use)."

Broker

An entity providing and/or connecting services on the behalf of other architectural components. Within the SACM Architecture, for example, a broker may provide authorization services and find, upon request, entities providing requested services.

Building Block
For SACM, a building block is a unit of functionality that may apply to more than one use case and can be supported by different components of an architectural model.

Capability

The extent of an architectural component’s ability. For example, a Posture Information Provider may only provide endpoint management data, and then only a subset of that data.

Client

An architectural component receiving services from another architectural component.

Collection Task

The process by which posture attributes or values are collected.

Consumer

An architectural component receiving information from another architectural component.

Endpoint

Defined in [RFC5209] as "any computing device that can be connected to a network. Such devices normally are associated with a particular link layer address before joining the network and potentially an IP address once on the network. This includes: laptops, desktops, servers, cell phones, or any device that may have an IP address."

To further clarify the [RFC5209] definition, an endpoint is any physical or virtual device that may have a network address. Note that, network infrastructure devices (e.g. switches, routers, firewalls), which fit the definition, are also considered to be endpoints within this document.

Based on the previous definition of an asset, an endpoint is a type of asset.

Evaluation Task

The process by which posture attributes are evaluated.

Endpoint Target
The endpoint of interest.

Endpoint Discovery

The process by which an endpoint can be identified.

Evaluation Result

The resulting value from having evaluated a set of posture attributes.

Expected Endpoint State

The required state of an endpoint that is to be compared against.

Function

A behavioral aspect of a particular architectural component, which belies that component’s purpose. For example, the Management Plane can provide a brokering function to other SACM architectural components.

Information Model

An information model is an abstract representation of data, their properties, relationships between data and the operations that can be performed on the data. While there is some overlap with a data model, [RFC3444] distinguished an information model as being protocol and implementation neutral whereas a data model would provide such details.

Management Plane (TBD per list; was "Control Plane")

Architectural component providing common functions to all SACM participants, including authentication, authorization, capabilities mappings, and the like.

Posture

Defined in [RFC5209] as "configuration and/or status of hardware or software on an endpoint as it pertains to an organization’s security policy."

This term is used within the scope of this document to represent the state information that is collected from an endpoint (e.g. software/hardware inventory, configuration settings). The state information may constitute one to many Posture Attributes.
Posture Attributes

Defined in [RFC5209] as "attributes describing the configuration or status (posture) of a feature of the endpoint. A Posture Attribute represents a single property of an observed state. For example, a Posture Attribute might describe the version of the operating system installed on the system."

Within this document this term represents a specific assertion about endpoint state (e.g. configuration setting, installed software, hardware). The phrase "features of the endpoint" refers to installed software or software components.

Provider

An architectural component providing information to another architectural component.

Proxy

An architectural component providing functions, information, or services on behalf of another component, which is not directly participating in the architecture.

Repository

An architectural component intended to store information of a particular kind. A single repository may provide the functions of more than one repository type (i.e. configuration baseline repository, assessment results repository, etc.)

Role

A label representing a collection of functions provided by a particular architectural component.

Security Automation

The process of which security alerts can be automated through the use of different tools to monitor, evaluate and analyze endpoint and network traffic for the purposes of detecting misconfigurations, misbehaviors or threats.

Supplicant

The entity seeking to be authenticated by the Management Plane for the purpose of participating in the SACM architecture.
System Resource

Defined in [RFC4949] as "data contained in an information system; or a service provided by a system; or a system capacity, such as processing power or communication bandwidth; or an item of system equipment (i.e., hardware, firmware, software, or documentation); or a facility that houses system operations and equipment."

3. IANA Considerations

This memo includes no request to IANA.

4. Security Considerations

This memo documents terminology for security automation. While it is about security, it does not affect security.

5. Acknowledgements

6. Change Log

6.1. ietf-sacm-terminology-01- to -02-

Added simple list of terms extracted from UC draft -05. It is expected that comments will be received on this list of terms as to whether they should be kept in this document. Those that are kept will be appropriately defined or cited.

6.2. ietf-sacm-terminology-01- to -02-

Added Vulnerability, Vulnerability Management, xposure, Misconfiguration, and Software flaw.

6.3. ietf-sacm-terminology-02- to -03-

Removed Section 2.1. Cleaned up some editing nits; broke terms into 2 sections (predefined and newly defined terms). Added some of the relevant terms per the proposed list discussed in the IETF 89 meeting.

6.4. ietf-sacm-terminology-03 to -04-

TODO
6.5. ietf-sacm-terminology-04 to -05-

TODO

6.6. ietf-sacm-terminology-05 to -06-

Updated author information.

Combined "Pre-defined Terms" with "New Terms and Definitions".

Removed "Requirements language".

Removed unused reference to use case draft; resulted in removal of normative references.

Removed introductory text from Section 1 indicating that this document is intended to be temporary.

Added placeholders for missing change log entries.

7. Informative References


Authors' Addresses

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

Email: david.waltermire@nist.gov
Internet-Draft     Terminology for Security Assessment     February 2015

Adam W. Montville
Center for Internet Security
31 Tech Valley Drive
East Greenbush, New York 12061
USA

Email: adam.w.montville@gmail.com

David Harrington
Effective Software
50 Harding Rd
Portsmouth, NH 03801
USA

Email: ietfdbh@comcast.net

Nancy Cam-Winget
Cisco Systems
3550 Cisco Way
San Jose, CA 95134
US

Email: ncamwing@cisco.com

Jarrett Lu
Oracle Corporation
4180 Network Circle
Santa Clara, California 95054

Email: jarrett.lu@oracle.com

Brian Ford
Lancope
3650 Brookside Parkway, Suite 500
Alpharetta, Georgia 30022

Email: bford@lancope.com
Abstract

This memo documents a sampling of use cases for securely aggregating configuration and operational data and evaluating that data to determine an organization’s security posture. From these operational use cases, we can derive common functional capabilities and requirements to guide development of vendor-neutral, interoperable standards for aggregating and evaluating data relevant to security posture.

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 September 25, 2015.

Copyright Notice

Copyright (c) 2015 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
Table of Contents

1.  Introduction .................................................. 2
2.  Endpoint Posture Assessment ................................. 4
   2.1.  Use Cases .................................................. 5
      2.1.1.  Define, Publish, Query and Retrieve Security
             Automation Data ..................................... 5
      2.1.2.  Endpoint Identification and Assessment Planning .... 9
      2.1.3.  Endpoint Posture Attribute Value Collection .......... 10
      2.1.4.  Posture Attribute Evaluation .......................... 11
   2.2.  Usage Scenarios ............................................ 12
      2.2.1.  Definition and Publication of Automatable
              Configuration Checklists ............................. 12
      2.2.2.  Automated Checklist Verification ...................... 13
      2.2.3.  Detection of Posture Deviations ....................... 16
      2.2.4.  Endpoint Information Analysis and Reporting .......... 17
      2.2.5.  Asynchronous Compliance/Vulnerability Assessment at
              Ice Station Zebra ..................................... 18
      2.2.6.  Identification and Retrieval of Guidance ............ 20
      2.2.7.  Guidance Change Detection ............................ 21
3.  IANA Considerations ............................................ 21
4.  Security Considerations ........................................ 21
5.  Acknowledgements ............................................... 22
6.  Change Log ..................................................... 22
   6.1.  -08- to -09- ............................................ 22
   6.2.  -07- to -08- ............................................ 22
   6.3.  -06- to -07- ............................................ 22
   6.4.  -05- to -06- ............................................ 23
   6.5.  -04- to -05- ............................................ 23
   6.6.  -03- to -04- ............................................ 24
   6.7.  -02- to -03- ............................................ 24
   6.8.  -01- to -02- ............................................ 25
   6.9.  -00- to -01- ............................................ 25
   6.10. draft-waltermire-sacm-use-cases-05 to draft-ietf-sacm-
        use-cases-00 .............................................. 26
   6.11. waltermire -04- to -05- ................................ 27
7.  Informative References ........................................ 28

1.  Introduction

This document describes the core set of use cases for endpoint
posture assessment for enterprises. It provides a discussion of
these use cases and associated building block capabilities. The described use cases support:

- securely collecting and aggregating configuration and operational data, and
- evaluating that data to determine the security posture of individual endpoints.

Additionally, this document describes a set of usage scenarios that provide examples for using the use cases and associated building blocks to address a variety of operational functions.

These operational use cases and related usage scenarios cross many IT security domains. The use cases enable the derivation of common:

- concepts that are expressed as building blocks in this document,
- characteristics to inform development of a requirements document
- information concepts to inform development of an information model document, and
- functional capabilities to inform development of an architecture document.

Together these ideas will be used to guide development of vendor-neutral, interoperable standards for collecting, aggregating, and evaluating data relevant to security posture.

Using this standard data, tools can analyze the state of endpoints, user activities and behaviour, and evaluate the security posture of an organization. Common expression of information should enable interoperability between tools (whether customized, commercial, or freely available), and the ability to automate portions of security processes to gain efficiency, react to new threats in a timely manner, and free up security personnel to work on more advanced problems.

The goal is to enable organizations to make informed decisions that support organizational objectives, to enforce policies for hardening systems, to prevent network misuse, to quantify business risk, and to collaborate with partners to identify and mitigate threats.

It is expected that use cases for enterprises and for service providers will largely overlap. When considering this overlap, there are additional complications for service providers, especially in handling information that crosses administrative domains.
The output of endpoint posture assessment is expected to feed into additional processes, such as policy-based enforcement of acceptable state, verification and monitoring of security controls, and compliance to regulatory requirements.

2. Endpoint Posture Assessment

Endpoint posture assessment involves orchestrating and performing data collection and evaluating the posture of a given endpoint. Typically, endpoint posture information is gathered and then published to appropriate data repositories to make collected information available for further analysis supporting organizational security processes.

Endpoint posture assessment typically includes:

- Collecting the attributes of a given endpoint;
- Making the attributes available for evaluation and action; and
- Verifying that the endpoint’s posture is in compliance with enterprise standards and policy.

As part of these activities, it is often necessary to identify and acquire any supporting security automation data that is needed to drive and feed data collection and evaluation processes.

The following is a typical workflow scenario for assessing endpoint posture:

1. Some type of trigger initiates the workflow. For example, an operator or an application might trigger the process with a request, or the endpoint might trigger the process using an event-driven notification.

2. An operator/application selects one or more target endpoints to be assessed.

3. An operator/application selects which policies are applicable to the targets.

4. For each target:
   
   A. The application determines which (sets of) posture attributes need to be collected for evaluation. Implementations should be able to support (possibly mixed) sets of standardized and proprietary attributes.
B. The application might retrieve previously collected information from a cache or data store, such as a data store populated by an asset management system.

C. The application might establish communication with the target, mutually authenticate identities and authorizations, and collect posture attributes from the target.

D. The application might establish communication with one or more intermediary/agents, mutually authenticate their identities and determine authorizations, and collect posture attributes about the target from the intermediary/agents. Such agents might be local or external.

E. The application communicates target identity and (sets of) collected attributes to an evaluator, possibly an external process or external system.

F. The evaluator compares the collected posture attributes with expected values as expressed in policies.

G. The evaluator reports the evaluation result for the requested assessment, in a standardized or proprietary format, such as a report, a log entry, a database entry, or a notification.

2.1. Use Cases

The following subsections detail specific use cases for assessment planning, data collection, analysis, and related operations pertaining to the publication and use of supporting data. Each use case is defined by a short summary containing a simple problem statement, followed by a discussion of related concepts, and a listing of associated building blocks which represent the capabilities needed to support the use case. These use cases and building blocks identify separate units of functionality that may be supported by different components of an architectural model.

2.1.1. Define, Publish, Query and Retrieve Security Automation Data

This use case describes the need for security automation data to be defined and published to one or more data stores, as well as queried and retrieved from these data stores for the explicit use of posture collection and evaluation.

Security automation data is a general concept that refers to any data expression that may be generated and/or used as part of the process of collecting and evaluating endpoint posture. Different types of
security automation data will generally fall into one of three categories:

Guidance: Instructions and related metadata that guide the attribute collection and evaluation processes. The purpose of this data is to allow implementations to be data-driven enabling their behavior to be customized without requiring changes to deployed software.

This type of data tends to change in units of months and days. In cases where assessments are made more dynamic, it may be necessary to handle changes in the scope of hours or minutes. This data will typically be provided by large organizations, product vendors, and some 3rd-parties. Thus, it will tend to be shared across large enterprises and customer communities. In some cases access may be controlled to specific authenticated users. In other cases, the data may be provided broadly with little to no access control.

This includes:

* Listings of attribute identifiers for which values may be collected and evaluated
* Lists of attributes that are to be collected along with metadata that includes: when to collect a set of attributes based on a defined interval or event, the duration of collection, and how to go about collecting a set of attributes.
* Guidance that specifies how old collected data can be to be used for evaluation.
* Policies that define how to target and perform the evaluation of a set of attributes for different kinds or groups of endpoints and the assets they are composed of. In some cases it may be desirable to maintain hierarchies of policies as well.
* References to human-oriented data that provide technical, organizational, and/or policy context. This might include references to: best practices documents, legal guidance and legislation, and instructional materials related to the automation data in question.

Attribute Data: Data collected through automated and manual mechanisms describing organizational and posture details pertaining to specific endpoints and the assets that they are
composed of (e.g., hardware, software, accounts). The purpose of this type of data is to characterize an endpoint (e.g., endpoint type, organizationally expected function/role) and to provide actual and expected state data pertaining to one or more endpoints. This data is used to determine what posture attributes to collect from which endpoints and to feed one or more evaluations.

This type of data tends to change in units of days, minutes, a seconds with posture attribute values typically changing more frequently than endpoint characterizations. This data tends to be organizationally and endpoint specific, with specific operational groups of endpoints tending to exhibit similar attribute profiles. This data will generally not be shared outside an organizational boundary and will generally require authentication with specific access controls.

This includes:

* Endpoint characterization data that describes the endpoint type, organizationally expected function/role, etc.

* Collected endpoint posture attribute values and related context including: time of collection, tools used for collection, etc.

* Organizationally defined expected posture attribute values targeted to specific evaluation guidance and endpoint characteristics. This allows a common set of guidance to be parameterized for use with different groups of endpoints.

Processing Artifacts: Data that is generated by, and is specific to, an individual assessment process. This data may be used as part of the interactions between architectural components to drive and coordinate collection and evaluation activities. Its lifespan will be bounded by the lifespan of the assessment. It may also be exchanged and stored to provide historic context around an assessment activity so that individual assessments can be grouped, evaluated, and reported in an enterprise context.

This includes:

* The identified set of endpoints for which an assessment should be performed.

* The identified set of posture attributes that need to be collected from specific endpoints to perform an evaluation.
The resulting data generated by an evaluation process including the context of what was assessed, what it was assessed against, what collected data was used, when it was collected, and when the evaluation was performed.

The information model for security automation data must support a variety of different data types as described above, along with the associated metadata that is needed to support publication, query, and retrieval operations. It is expected that multiple data models will be used to express specific data types requiring specialized or extensible security automation data repositories. The different temporal characteristics, access patterns, and access control dimensions of each data type may also require different protocols and data models to be supported furthering the potential requirement for specialized data repositories. See [RFC3444] for a description and discussion of distinctions between an information and data model. It is likely that additional kinds of data will be identified through the process of defining requirements and an architectural model. Implementations supporting this building block will need to be extensible to accommodate the addition of new types of data, both proprietary or (preferably) using a standard format.

The building blocks of this use case are:

Data Definition: Security automation data will guide and inform collection and evaluation processes. This data may be designed by a variety of roles - application implementers may build security automation data into their applications; administrators may define guidance based on organizational policies; operators may define guidance and attribute data as needed for evaluation at runtime, and so on. Data producers may choose to reuse data from existing stores of security automation data and/or may create new data. Data producers may develop data based on available standardized or proprietary data models, such as those used for network management and/or host management.

Data Publication: The capability to enable data producers to publish data to a security automation data store for further use. Published data may be made publicly available or access may be based on an authorization decision using authenticated credentials. As a result, the visibility of specific security automation data to an operator or application may be public, enterprise-scoped, private, or controlled within any other scope.

Data Query: An operator or application should be able to query a security automation data store using a set of specified
criteria. The result of the query will be a listing matching the query. The query result listing may contain publication metadata (e.g., create date, modified date, publisher, etc.) and/or the full data, a summary, snippet, or the location to retrieve the data.

Data Retrieval: A user, operator, or application acquires one or more specific security automation data entries. The location of the data may be known a priori, or may be determined based on decisions made using information from a previous query.

Data Change Detection: An operator or application needs to know when security automation data they are interested in has been published to, updated in, or deleted from a security automation data store which they have been authorized to access.

These building blocks are used to enable acquisition of various instances of security automation data based on specific data models that are used to drive assessment planning (see section 2.1.2), posture attribute value collection (see section 2.1.3), and posture evaluation (see section 2.1.4).

2.1.2. Endpoint Identification and Assessment Planning

This use case describes the process of discovering endpoints, understanding their composition, identifying the desired state to assess against, and calculating what posture attributes to collect to enable evaluation. This process may be a set of manual, automated, or hybrid steps that are performed for each assessment.

The building blocks of this use case are:

Endpoint Discovery: To determine the current or historic presence of endpoints in the environment that are available for posture assessment. Endpoints are identified in support of discovery using information previously obtained or by using other collection mechanisms to gather identification and characterization data. Previously obtained data may originate from sources such as network authentication exchanges.

Endpoint Characterization: The act of acquiring, through automated collection or manual input, and organizing attributes associated with an endpoint (e.g., type, organizationally expected function/role, hardware/software versions).

Identify Endpoint Targets: Determine the candidate endpoint target(s) against which to perform the assessment. Depending on the assessment trigger, a single endpoint or multiple
endpoints may be targeted based on characterized endpoint attributes. Guidance describing the assessment to be performed may contain instructions or references used to determine the applicable assessment targets. In this case the Data Query and/or Data Retrieval building blocks (see section 2.1.1) may be used to acquire this data.

Endpoint Component Inventory: To determine what applicable desired states should be assessed, it is first necessary to acquire the inventory of software, hardware, and accounts associated with the targeted endpoint(s). If the assessment of the endpoint is not dependent on these details, then this capability is not required for use in performing the assessment. This process can be treated as a collection use case for specific posture attributes. In this case the building blocks for Endpoint Posture Attribute Value Collection (see section 2.1.3) can be used.

Posture Attribute Identification: Once the endpoint targets and their associated asset inventory is known, it is then necessary to calculate what posture attributes are required to be collected to perform the desired evaluation. When available, existing posture data is queried for suitability using the Data Query building block (see section 2.1.1). Such posture data is suitable if it is complete and current enough for use in the evaluation. Any unsuitable posture data is identified for collection.

If this is driven by guidance, then the Data Query and/or Data Retrieval building blocks (see section 2.1.1) may be used to acquire this data.

At this point the set of posture attribute values to use for evaluation are known and they can be collected if necessary (see section 2.1.3).

2.1.3. Endpoint Posture Attribute Value Collection

This use case describes the process of collecting a set of posture attribute values related to one or more endpoints. This use case can be initiated by a variety of triggers including:

1. A posture change or significant event on the endpoint.

2. A network event (e.g., endpoint connects to a network/VPN, specific netflow is detected).

3. A scheduled or ad hoc collection task.
The building blocks of this use case are:

Collection Guidance Acquisition: If guidance is required to drive the collection of posture attributes values, this capability is used to acquire this data from one or more security automation data stores. Depending on the trigger, the specific guidance to acquire might be known. If not, it may be necessary to determine the guidance to use based on the component inventory or other assessment criteria. The Data Query and/or Data Retrieval building blocks (see section 2.1.1) may be used to acquire this guidance.

Posture Attribute Value Collection: The accumulation of posture attribute values. This may be based on collection guidance that is associated with the posture attributes.

Once the posture attribute values are collected, they may be persisted for later use or they may be immediately used for posture evaluation.

2.1.4. Posture Attribute Evaluation

This use case represents the action of analyzing collected posture attribute values as part of an assessment. The primary focus of this use case is to support evaluation of actual endpoint state against the expected state selected for the assessment.

This use case can be initiated by a variety of triggers including:

1. A posture change or significant event on the endpoint.

2. A network event (e.g., endpoint connects to a network/VPN, specific netflow is detected).

3. A scheduled or ad hoc evaluation task.

The building blocks of this use case are:

Collected Posture Change Detection: An operator or application has a mechanism to detect the availability of new, or changes to existing, posture attribute values. The timeliness of detection may vary from immediate to on-demand. Having the ability to filter what changes are detected will allow the operator to focus on the changes that are relevant to their use and will enable evaluation to occur dynamically based on detected changes.
Posture Attribute Value Query: If previously collected posture attribute values are needed, the appropriate data stores are queried to retrieve them using the Data Query building block (see section 2.1.1). If all posture attribute values are provided directly for evaluation, then this capability may not be needed.

Evaluation Guidance Acquisition: If guidance is required to drive the evaluation of posture attributes values, this capability is used to acquire this data from one or more security automation data stores. Depending on the trigger, the specific guidance to acquire might be known. If not, it may be necessary to determine the guidance to use based on the component inventory or other assessment criteria. The Data Query and/or Data Retrieval building blocks (see section 2.1.1) may be used to acquire this guidance.

Posture Attribute Evaluation: The comparison of posture attribute values against their expected values as expressed in the specified guidance. The result of this comparison is output as a set of posture evaluation results. Such results include metadata required to provide a level of assurance with respect to the posture attribute data and, therefore, evaluation results. Examples of such metadata include provenance and or availability data.

While the primary focus of this use case is around enabling the comparison of expected vs. actual state, the same building blocks can support other analysis techniques that are applied to collected posture attribute data (e.g., trending, historic analysis).

Completion of this process represents a complete assessment cycle as defined in Section 2.

2.2. Usage Scenarios

In this section, we describe a number of usage scenarios that utilize aspects of endpoint posture assessment. These are examples of common problems that can be solved with the building blocks defined above.

2.2.1. Definition and Publication of Automatable Configuration Checklists

A vendor manufactures a number of specialized endpoint devices. They also develop and maintain an operating system for these devices that enables end-user organizations to configure a number of security and operational settings. As part of their customer support activities,
they publish a number of secure configuration guides that provide
minimum security guidelines for configuring their devices.

Each guide they produce applies to a specific model of device and
version of the operating system and provides a number of specialized
configurations depending on the device’s intended function and what
add-on hardware modules and software licenses are installed on the
device. To enable their customers to evaluate the security posture
of their devices to ensure that all appropriate minimal security
settings are enabled, they publish an automatable configuration
checklists using a popular data format that defines what settings to
collect using a network management protocol and appropriate values
for each setting. They publish these checklists to a public security
automation data store that customers can query to retrieve applicable
checklist(s) for their deployed specialized endpoint devices.

Automatable configuration checklist could also come from sources
other than a device vendor, such as industry groups or regulatory
authorities, or enterprises could develop their own checklists.

This usage scenario employs the following building blocks defined in
Section 2.1.1 above:

Data Definition: To allow guidance to be defined using standardized
or proprietary data models that will drive collection and
evaluation.

Data Publication: Providing a mechanism to publish created guidance
to a security automation data store.

Data Query: To locate and select existing guidance that may be
reused.

Data Retrieval To retrieve specific guidance from a security
automation data store for editing.

While each building block can be used in a manual fashion by a human
operator, it is also likely that these capabilities will be
implemented together in some form of a guidance editor or generator
application.

2.2.2. Automated Checklist Verification

A financial services company operates a heterogeneous IT environment.
In support of their risk management program, they utilize vendor
provided automatable security configuration checklists for each
operating system and application used within their IT environment.
Multiple checklists are used from different vendors to insure adequate coverage of all IT assets.

To identify what checklists are needed, they use automation to gather an inventory of the software versions utilized by all IT assets in the enterprise. This data gathering will involve querying existing data stores of previously collected endpoint software inventory posture data and actively collecting data from reachable endpoints as needed utilizing network and systems management protocols. Previously collected data may be provided by periodic data collection, network connection-driven data collection, or ongoing event-driven monitoring of endpoint posture changes.

Appropriate checklists are queried, located and downloaded from the relevant guidance data stores. The specific data stores queried and the specifics of each query may be driven by data including:

- collected hardware and software inventory data, and
- associated asset characterization data that may indicate the organizational defined functions of each endpoint.

Checklists may be sourced from guidance data stores maintained by an application or OS vendor, an industry group, a regulatory authority, or directly by the enterprise.

The retrieved guidance is cached locally to reduce the need to retrieve the data multiple times.

Driven by the setting data provided in the checklist, a combination of existing configuration data stores and data collection methods are used to gather the appropriate posture attributes from (or pertaining to) each endpoint. Specific posture attribute values are gathered based on the defined enterprise function and software inventory of each endpoint. The collection mechanisms used to collect software inventory posture will be used again for this purpose. Once the data is gathered, the actual state is evaluated against the expected state criteria defined in each applicable checklist.

A checklist can be assessed as a whole, or a specific subset of the checklist can be assessed resulting in partial data collection and evaluation.

The results of checklist evaluation are provided to appropriate operators and applications to drive additional business logic. Specific applications for checklist evaluation results are out-of-scope for current SACM efforts. Irrespective of specific applications, the availability, timeliness, and liveness of results
is often of general concern. Network latency and available bandwidth often create operational constraints that require trade-offs between these concerns and need to be considered.

Uses of checklists and associated evaluation results may include, but are not limited to:

- Detecting endpoint posture deviations as part of a change management program to:
  * identify missing required patches,
  * unauthorized changes to hardware and software inventory, and
  * unauthorized changes to configuration items.

- Determining compliance with organizational policies governing endpoint posture.

- Informing configuration management, patch management, and vulnerability mitigation and remediation decisions.

- Searching for current and historic indicators of compromise.

- Detecting current and historic infection by malware and determining the scope of infection within an enterprise.

- Detecting performance, attack and vulnerable conditions that warrant additional network diagnostics, monitoring, and analysis.

- Informing network access control decision making for wired, wireless, or VPN connections.

This usage scenario employs the following building blocks defined in Section 2.1.1 above:

Endpoint Discovery: The purpose of discovery is to determine the type of endpoint to be posture assessed.

Identify Endpoint Targets: To identify what potential endpoint targets the checklist should apply to based on organizational policies.

Endpoint Component Inventory: Collecting and consuming the software and hardware inventory for the target endpoints.

Posture Attribute Identification: To determine what data needs to be collected to support evaluation, the checklist is evaluated.
against the component inventory and other endpoint metadata to determine the set of posture attribute values that are needed.

Collection Guidance Acquisition: Based on the identified posture attributes, the application will query appropriate security automation data stores to find the "applicable" collection guidance for each endpoint in question.

Posture Attribute Value Collection: For each endpoint, the values for the required posture attributes are collected.

Posture Attribute Value Query: If previously collected posture attribute values are used, they are queried from the appropriate data stores for the target endpoint(s).

Evaluation Guidance Acquisition: Any guidance that is needed to support evaluation is queried and retrieved.

Posture Attribute Evaluation: The resulting posture attribute values from previous collection processes are evaluated using the evaluation guidance to provide a set of posture results.

2.2.3. Detection of Posture Deviations

Example corporation has established secure configuration baselines for each different type of endpoint within their enterprise including: network infrastructure, mobile, client, and server computing platforms. These baselines define an approved list of hardware, software (i.e., operating system, applications, and patches), and associated required configurations. When an endpoint connects to the network, the appropriate baseline configuration is communicated to the endpoint based on its location in the network, the expected function of the device, and other asset management data. It is checked for compliance with the baseline indicating any deviations to the device’s operators. Once the baseline has been established, the endpoint is monitored for any change events pertaining to the baseline on an ongoing basis. When a change occurs to posture defined in the baseline, updated posture information is exchanged, allowing operators to be notified and/or automated action to be taken.

Like the Automated Checklist Verification usage scenario (see section 2.2.2), this usage scenario supports assessment based on automatable checklists. It differs from that scenario by monitoring for specific endpoint posture changes on an ongoing basis. When the endpoint detects a posture change, an alert is generated identifying the specific changes in posture allowing assessment of the delta to be performed instead of a full assessment in the previous case. This
usage scenario employs the same building blocks as Automated Checklist Verification (see section 2.2.2). It differs slightly in how it uses the following building blocks:

Endpoint Component Inventory: Additionally, changes to the hardware and software inventory are monitored, with changes causing alerts to be issued.

Posture Attribute Value Collection: After the initial assessment, posture attributes are monitored for changes. If any of the selected posture attribute values change, an alert is issued.

Posture Attribute Value Query: The previous state of posture attributes are tracked, allowing changes to be detected.

Posture Attribute Evaluation: After the initial assessment, a partial evaluation is performed based on changes to specific posture attributes.

This usage scenario highlights the need to query a data store to prepare a compliance report for a specific endpoint and also the need for a change in endpoint state to trigger Collection and Evaluation.

2.2.4. Endpoint Information Analysis and Reporting

Freed from the drudgery of manual endpoint compliance monitoring, one of the security administrators at Example Corporation notices (not using SACM standards) that five endpoints have been uploading lots of data to a suspicious server on the Internet. The administrator queries data stores for specific endpoint posture to see what software is installed on those endpoints and finds that they all have a particular program installed. She then queries the appropriate data stores to see which other endpoints have that program installed. All these endpoints are monitored carefully (not using SACM standards), which allows the administrator to detect that the other endpoints are also infected.

This is just one example of the useful analysis that a skilled analyst can do using data stores of endpoint posture.

This usage scenario employs the following building blocks defined in Section 2.1.1 above:

Posture Attribute Value Query: Previously collected posture attribute values for the target endpoint(s) are queried from the appropriate data stores using a standardized method.
This usage scenario highlights the need to query a repository for attributes to see which attributes certain endpoints have in common.

2.2.5. Asynchronous Compliance/Vulnerability Assessment at Ice Station Zebra

A university team receives a grant to do research at a government facility in the arctic. The only network communications will be via an intermittent, low-speed, high-latency, high-cost satellite link. During their extended expedition, they will need to show continue compliance with the security policies of the university, the government, and the provider of the satellite network as well as keep current on vulnerability testing. Interactive assessments are therefore not reliable, and since the researchers have very limited funding they need to minimize how much money they spend on network data.

Prior to departure they register all equipment with an asset management system owned by the university, which will also initiate and track assessments.

On a periodic basis -- either after a maximum time delta or when the security automation data store has received a threshold level of new vulnerability definitions -- the university uses the information in the asset management system to put together a collection request for all of the deployed assets that encompasses the minimal set of artifacts necessary to evaluate all three security policies as well as vulnerability testing.

In the case of new critical vulnerabilities, this collection request consists only of the artifacts necessary for those vulnerabilities and collection is only initiated for those assets that could potentially have a new vulnerability.

(Optional) Asset artifacts are cached in a local CMDB. When new vulnerabilities are reported to the security automation data store, a request to the live asset is only done if the artifacts in the CMDB are incomplete and/or not current enough.

The collection request is queued for the next window of connectivity. The deployed assets eventually receive the request, fulfill it, and queue the results for the next return opportunity.

The collected artifacts eventually make it back to the university where the level of compliance and vulnerability exposed is calculated and asset characteristics are compared to what is in the asset management system for accuracy and completeness.
Like the Automated Checklist Verification usage scenario (see section 2.2.2), this usage scenario supports assessment based on checklists. It differs from that scenario in how guidance, collected posture attribute values, and evaluation results are exchanged due to bandwidth limitations and availability. This usage scenario employs the same building blocks as Automated Checklist Verification (see section 2.2.2). It differs slightly in how it uses the following building blocks:

Endpoint Component Inventory: It is likely that the component inventory will not change. If it does, this information will need to be batched and transmitted during the next communication window.

Collection Guidance Acquisition: Due to intermittent communication windows and bandwidth constraints, changes to collection guidance will need to be batched and transmitted during the next communication window. Guidance will need to be cached locally to avoid the need for remote communications.

Posture Attribute Value Collection: The specific posture attribute values to be collected are identified remotely and batched for collection during the next communication window. If a delay is introduced for collection to complete, results will need to be batched and transmitted.

Posture Attribute Value Query: Previously collected posture attribute values will be stored in a remote data store for use at the university.

Evaluation Guidance Acquisition: Due to intermittent communication windows and bandwidth constraints, changes to evaluation guidance will need to be batched and transmitted during the next communication window. Guidance will need to be cached locally to avoid the need for remote communications.

Posture Attribute Evaluation: Due to the caching of posture attribute values and evaluation guidance, evaluation may be performed at both the university campus as well as the satellite site.

This usage scenario highlights the need to support low-bandwidth, intermittent, or high-latency links.
2.2.6. Identification and Retrieval of Guidance

In preparation for performing an assessment, an operator or application will need to identify one or more security automation data stores that contain the guidance entries necessary to perform data collection and evaluation tasks. The location of a given guidance entry will either be known a priori or known security automation data stores will need to be queried to retrieve applicable guidance.

To query guidance it will be necessary to define a set of search criteria. This criteria will often utilize a logical combination of publication metadata (e.g. publishing identity, create time, modification time) and guidance data-specific criteria elements. Once the criteria is defined, one or more security automation data stores will need to be queried generating a result set. Depending on how the results are used, it may be desirable to return the matching guidance directly, a snippet of the guidance matching the query, or a resolvable location to retrieve the data at a later time. The guidance matching the query will be restricted based the authorized level of access allowed to the requester.

If the location of guidance is identified in the query result set, the guidance will be retrieved when needed using one or more data retrieval requests. A variation on this approach would be to maintain a local cache of previously retrieved data. In this case, only guidance that is determined to be stale by some measure will be retrieved from the remote data store.

Alternately, guidance can be discovered by iterating over data published with a given context within a security automation data store. Specific guidance can be selected and retrieved as needed.

This usage scenario employs the following building blocks defined in Section 2.1.1 above:

Data Query: Enables an operator or application to query one or more security automation data stores for guidance using a set of specified criteria.

Data Retrieval: If data locations are returned in the query result set, then specific guidance entries can be retrieved and possibly cached locally.
2.2.7. Guidance Change Detection

An operator or application may need to identify new, updated, or deleted guidance in a security automation data store for which they have been authorized to access. This may be achieved by querying or iterating over guidance in a security automation data store, or through a notification mechanism that alerts to changes made to a security automation data store.

Once guidance changes have been determined, data collection and evaluation activities may be triggered.

This usage scenario employs the following building blocks defined in Section 2.1.1 above:

Data Change Detection: Allows an operator or application to identify guidance changes in a security automation data store which they have been authorized to access.

Data Retrieval: If data locations are provided by the change detection mechanism, then specific guidance entries can be retrieved and possibly cached locally.

3. IANA Considerations

This memo includes no request to IANA.

4. Security Considerations

This memo documents, for informational purposes, use cases for security automation. Specific security considerations will be provided in related documents (e.g., requirements, architecture, information model, data model, protocol) as appropriate to the function described in each related document.

One consideration for security automation is that a malicious actor could use the security automation infrastructure and related collected data to determine endpoint weaknesses to exploit. It is important that security considerations in the related documents identify methods to both identify and prevent such activity. Specifically, means for protecting the communications as well as the systems that store the information. For communications between the varying SACM components there should be considerations for protecting the confidentiality, data integrity and peer entity authentication. Also, for any systems that store information that could be used for malicious purposes, methods to identify and protect against unauthorized usage, inappropriate usage and denial of service need to be considered.
5. Acknowledgements

Adam Montville edited early versions of this draft.

Kathleen Moriarty, and Stephen Hanna contributed text describing the scope of the document.

Gunnar Engelbach, Steve Hanna, Chris Inacio, Kent Landfield, Lisa Lorenzin, Adam Montville, Kathleen Moriarty, Nancy Cam-Winget, and Aron Woland provided use cases text for various revisions of this draft.

6. Change Log

6.1. -08- to -09-

Fixed a number of grammatical nits throughout the draft identified by the SECDIR review.

Added additional text to the security considerations about malicious actors.

6.2. -07- to -08-

Reworked long sentences throughout the document by shortening or using bulleted lists.

Re-ordered and condensed text in the "Automated Checklist Verification" sub-section to improve the conceptual presentation and to clarify longer sentences.

Clarified that the "Posture Attribute Value Query" building block represents a standardized interface in the context of SACM.

Removed the "others" sub-section within the "usage scenarios" section.

Updated the "Security Considerations" section to identify that actual SACM security considerations will be discussed in the appropriate related documents.

6.3. -06- to -07-

A number of edits were made to section 2 to resolve open questions in the draft based on meeting and mailing list discussions.

Section 2.1.5 was merged into section 2.1.4.
6.4. -05- to -06-

Updated the "Introduction" section to better reflect the use case, building block, and usage scenario structure changes from previous revisions.

Updated most uses of the terms "content" and "content repository" to use "guidance" and "security automation data store" respectively.

In section 2.1.1, added a discussion of different data types and renamed "content" to "data" in the building block names.

In section 2.1.2, separated out the building block concepts of "Endpoint Discovery" and "Endpoint Characterization" based on mailing list discussions.

Addressed some open questions throughout the draft based on consensus from mailing list discussions and the two virtual interim meetings.

Changed many section/sub-section names to better reflect their content.

6.5. -04- to -05-

Changes in this revision are focused on section 2 and the subsequent subsections:

- Moved existing use cases to a subsection titled "Usage Scenarios".
- Added a new subsection titled "Use Cases" to describe the common use cases and building blocks used to address the "Usage Scenarios". The new use cases are:
  * Define, Publish, Query and Retrieve Content
  * Endpoint Identification and Assessment Planning
  * Endpoint Posture Attribute Value Collection
  * Posture Evaluation
  * Mining the Database
- Added a listing of building blocks used for all usage scenarios.
- Combined the following usage scenarios into "Automated Checklist Verification": "Organizational Software Policy Compliance", "Search for Signs of Infection", "Vulnerable Endpoint
o Created new usage scenario "Identification and Retrieval of Repository Content" by combining the following usage scenarios: "Repository Interaction - A Full Assessment" and "Repository Interaction - Filtered Delta Assessment"

o Renamed "Register with repository for immediate notification of new security vulnerability content that match a selection filter" to "Content Change Detection" and generalized the description to be neutral to implementation approaches.

o Removed out-of-scope usage scenarios: "Remediation and Mitigation" and "Direct Human Retrieval of Ancillary Materials"

Updated acknowledgements to recognize those that helped with editing the use case text.

6.6. -03- to -04-

Added four new use cases regarding content repository.

6.7. -02- to -03-

Expanded the workflow description based on ML input.

Changed the ambiguous "assess" to better separate data collection from evaluation.

Added use case for Search for Signs of Infection.

Added use case for Remediation and Mitigation.

Added use case for Endpoint Information Analysis and Reporting.

Added use case for Asynchronous Compliance/Vulnerability Assessment at Ice Station Zebra.

Added use case for Traditional endpoint assessment with stored results.

Added use case for NAC/NAP connection with no stored results using an endpoint evaluator.
Added use case for NAC/NAP connection with no stored results using a third-party evaluator.

Added use case for Compromised Endpoint Identification.

Added use case for Suspicious Endpoint Behavior.

Added use case for Vulnerable Endpoint Identification.

Updated Acknowledgements

6.8. -01- to -02-

Changed title

removed section 4, expecting it will be moved into the requirements document.

removed the list of proposed capabilities from section 3.1

Added empty sections for Search for Signs of Infection, Remediation and Mitigation, and Endpoint Information Analysis and Reporting.

Removed Requirements Language section and rfc2119 reference.

Removed unused references (which ended up being all references).

6.9. -00- to -01-

o Work on this revision has been focused on document content relating primarily to use of asset management data and functions.

o Made significant updates to section 3 including:

* Reworked introductory text.

* Replaced the single example with multiple use cases that focus on more discrete uses of asset management data to support hardware and software inventory, and configuration management use cases.

* For one of the use cases, added mapping to functional capabilities used. If popular, this will be added to the other use cases as well.

* Additional use cases will be added in the next revision capturing additional discussion from the list.
Made significant updates to section 4 including:

* Renamed the section heading from "Use Cases" to "Functional Capabilities" since use cases are covered in section 3. This section now extrapolates specific functions that are needed to support the use cases.

* Started work to flatten the section, moving select subsections up from under asset management.

* Removed the subsections for: Asset Discovery, Endpoint Components and Asset Composition, Asset Resources, and Asset Life Cycle.

* Renamed the subsection "Asset Representation Reconciliation" to "Deconfliction of Asset Identities".

* Expanded the subsections for: Asset Identification, Asset Characterization, and Deconfliction of Asset Identities.

* Added a new subsection for Asset Targeting.

* Moved remaining sections to "Other Unedited Content" for future updating.

6.10. draft-waltermire-sacm-use-cases-05 to draft-ietf-sacm-use-cases-00

* Transitioned from individual I/D to WG I/D based on WG consensus call.

* Fixed a number of spelling errors. Thank you Erik!

* Added keywords to the front matter.

* Removed the terminology section from the draft. Terms have been moved to: draft-dbh-sacm-terminology-00

* Removed requirements to be moved into a new I/D.

* Extracted the functionality from the examples and made the examples less prominent.

* Renamed "Functional Capabilities and Requirements" section to "Use Cases".

  * Reorganized the "Asset Management" sub-section. Added new text throughout.
+ Renamed a few sub-section headings.
+ Added text to the "Asset Characterization" sub-section.

o Renamed "Security Configuration Management" to "Endpoint Configuration Management". Not sure if the "security" distinction is important.

* Added new sections, partially integrated existing content.
* Additional text is needed in all of the sub-sections.

o Changed "Security Change Management" to "Endpoint Posture Change Management". Added new skeletal outline sections for future updates.

6.11. waltermire -04- to -05-

o Are we including user activities and behavior in the scope of this work? That seems to be layer 8 stuff, appropriate to an IDS/IPS application, not Internet stuff.

o Removed the references to what the WG will do because this belongs in the charter, not the (potentially long-lived) use cases document. I removed mention of charter objectives because the charter may go through multiple iterations over time; there is a website for hosting the charter; this document is not the correct place for that discussion.

o Moved the discussion of NIST specifications to the acknowledgements section.

o Removed the portion of the introduction that describes the chapters; we have a table of concepts, and the existing text seemed redundant.

o Removed marketing claims, to focus on technical concepts and technical analysis, that would enable subsequent engineering effort.

o Removed (commented out in XML) UC2 and UC3, and eliminated some text that referred to these use cases.

o Modified IANA and Security Consideration sections.

o Moved Terms to the front, so we can use them in the subsequent text.
o Removed the "Key Concepts" section, since the concepts of ORM and IRM were not otherwise mentioned in the document. This would seem more appropriate to the arch doc rather than use cases.

o Removed role=editor from David Waltermire’s info, since there are three editors on the document. The editor is most important when one person writes the document that represents the work of multiple people. When there are three editors, this role marking isn't necessary.

o Modified text to describe that this was specific to enterprises, and that it was expected to overlap with service provider use cases, and described the context of this scoped work within a larger context of policy enforcement, and verification.

o The document had asset management, but the charter mentioned asset, change, configuration, and vulnerability management, so I added sections for each of those categories.

o Added text to Introduction explaining goal of the document.

o Added sections on various example use cases for asset management, config management, change management, and vulnerability management.

7. Informative References


Authors’ Addresses

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

Email: david.waltermire@nist.gov